ＣＳＣ９０１ＮＴ基本セット
ＣＳＣ９０２基本セット
ＣＳＣ９０３基本セット
FV904基本ｾｯﾄ　FVL-BIS/904

９０Ｘ
ホスト通信
ライブラリ説明書
☆第３版☆

（株）ファースト

御注意

◎本書の内容の一部または全部を無断で転載することは固くお断りします。
◎本書の内容について、将来改良を目的に予告なしに変更することがあります。
◎Turbo C++は、Borland Internationalの商標です。
◎Microsoft Cは、Microsoft Corp.の商標です。
◎Watcom C™は、Watcom International Corporationの商標です。
◎Visual C++は、Microsoft Corp.の商標です。
◎MS-Windows95/NTは、Microsoft Corp.の商標です。
◎WinRTは、BlueWaterSystem Corp.の商標です。

はじめに

この説明書は、以下のバージョン以降に対応しています。

ＣＳＣ９０１システム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ Ver.6.40
ＣＳＣ９０２システム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ Ver.2.50
ＣＳＣ９０３システム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ Ver.1.10
ＣＳＣ９０４システム ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・ Ver.1.00
ＤＯＳ版　ホストＣ－ＮＥＴライブラリ ・・・・・・・・・・・・・・・・・・ Ver.2.30
ＤＯＳ版　ホストＣ－ＬＩＮＫライブラリ ・・・・・・・・・・・・・・・・・ Ver.1.40
ＤＯＳ版　ホストパラレル・リンクライブラリ ・・・・・・・・・・・・・ Ver.1.10
Ｗｉｎｄｏｗｓ９５版　ホスト通信ライブラリ ・・・・・・・・・・・・・ Ver.1.40
ＷｉｎｄｏｗｓＮＴ版　ホスト通信ライブラリ ・・・・・・・・・・・・・ Ver.1.40

本書は、スーパービジョン９０Ｘのホスト通信ライブラリについて記載したものです。
なお、下記の説明書も併せてご参照ください。

◎９０Ｘ 操作説明書
◎ＰＣ９８　Ｃ－ＮＥＴ　Ｉ／Ｆボード取扱説明書
◎ＰＣ／ＡＴ　Ｃ－ＮＥＴ　Ｉ／Ｆボード取扱説明書

Ｃ－ＮＥＴを使用する際の注意事項

ＰＣ－９８００シリーズをご使用のお客様へ

サウンド機能が内蔵されているＰＣ－９８００シリーズでＣ－ＮＥＴを使用する場合、サウンド機能の割り込みレベル（ＩＮＴ５）とＣ－ＮＥＴの工場出荷時の割り込みレベルが衝突する可能性があります。

この場合は、Ｃ－ＮＥＴの割り込みレベルを変更するか、ＰＣ－９８００シリーズの内蔵サウンド機能を切り放す必要があります。

Ｃ－ＮＥＴの割り込みレベルの変更の仕方は、

“**ＰＣ９８　Ｃ－ＮＥＴ　Ｉ／Ｆボード　取扱説明書**”

をご覧ください。

ＰＣ－９８００シリーズの内蔵サウンド機能の切り放しの仕方

① ［ＨＥＬＰ］ キーを押しながら電源をＯＮにします。

② ［↑］ ［↓］ キーを押して反転表示を上下に動かしてメインメニューから動作環境の設定を選択し、［↵］ キーを押します。

③ ［↑］ ［↓］ キーでサウンドの項目を選択し、［→］ ［←］ キーで設定値を“使用しない”に設定します。

④ ［ＥＳＣ］ キーを押すか、“終了”の項目を選択し［↵］ キーを押します。

⑤メインメニューを④と同じ方法で終了し、再起動します。

インストール

この項は、ホスト通信ライブラリのインストールについて書かれています。

Ｗｉｎｄｏｗｓ９５／ＮＴからのインストール

「ＣＳＣ９０Ｘ　ホスト通信ライブラリＦＤ　＃１」をフロッピードライブに挿入し、カレントドライブをフロッピードライブに変更します。

```
A＞ＳＥＴＵＰ
```

と入力します。
但し、インストールを実行するには１Ｍバイト以上のハードディスク容量が必要です。
インストールプログラムが立ち上がりましたら、コンソールに表示されるメッセージに従って操作してください。

ＤＯＳからのコピー

ＤＯＳからは以下のライブラリをコピーすることができます。

ＤＯＳ１６ビット版　ホストＣ－ＮＥＴライブラリ
ＤＯＳ１６ビット版　ホストＣ－ＬＩＮＫライブラリ
ＤＯＳ１６ビット版　ホストパラレル・リンクライブラリ
ＤＯＳ３２ビット版　ホストＣ－ＮＥＴライブラリ
ＤＯＳ３２ビット版　ホストＣ－ＬＩＮＫライブラリ
ＤＯＳ３２ビット版　ホストパラレル・リンクライブラリ

「ＣＳＣ９０Ｘ　ホスト通信ライブラリＦＤ　＃３」の￥ＤＯＳディレクトリは以下のような構成になっています。

<DOS> <LIB16> <C_NET>・・・・ＤＯＳ１６ビット版　ホストＣ－ＮＥＴライブラリ
<DOS> <LIB16> <C_LINK>・・・ＤＯＳ１６ビット版　ホストＣ－ＬＩＮＫライブラリ
<DOS> <LIB16> <P_LINK>・・・ＤＯＳ１６ビット版　ホストパラレル・リンクライブラリ
<DOS> <LIB32> <C_NET>・・・・ＤＯＳ３２ビット版　ホストＣ－ＮＥＴライブラリ
<DOS> <LIB32> <C_LINK>・・・ＤＯＳ３２ビット版　ホストＣ－ＬＩＮＫライブラリ
<DOS> <LIB32> <P_LINK>・・・ＤＯＳ３２ビット版　ホストパラレル・リンクライブラリ

必要なライブラリをコピーしてください。

※Ｗｉｎｄｏｗｓ９５／ＮＴ版　ホスト通信ライブラリは、ＤＯＳからコピーすることはできません。

目　次

第１編　ＤＯＳ版　ホストＣ－ＮＥＴライブラリ

１．概　　要 ・・・

目　　次

第２編　ＤＯＳ版　ホストＣ－ＬＩＮＫライブラリ

１．概　　要 ・・・

目　次

第３編　ＤＯＳ版　ホストパラレル・リンクライブラリ

１．概　　要 ・・・

目　　次

第４編　Windows95/NT 版　ホスト通信ライブラリ

１．概　　要 ・・・

第１編

ＤＯＳ版
ホストＣ－ＮＥＴライブラリ

アルファベット順ライブラリ目次

項番	ライブラリ名	ページ
1	C_NET_Finish	14
2	C_NET_Get_Node_Map	18
3	C_NET_Init	10
4	C_NET_IRQ_Clear	17
5	C_NET_IRQ_Set	15
6	C_NET_Receive	21
7	C_NET_Receive_Status	20
8	C_NET_Send	24
9	C_NET_Send_Status	22
10	checkccmdack	32
11	checkcprmpt	33
12	cload	26
13	clrcnetrcv	41
14	cnetbuffcheck	45
15	cnetbuffclose	39
16	cnetbuffclr	47
17	cnetbuffopen	38
18	cnetbuffrcv	46
19	cnetbuffstatus	49
20	cupload	28
21	getcnetrcv	43
22	gets_cnet	36
23	outccmd	30
24	puts_cnet	34
25	setcnetrcv	40

１．概　　要

本ライブラリは、ＣＳＣ９０Ｘとホスト間をＣ－ＮＥＴによる通信を行うライブラリです。
本ライブラリには、ＭＳ－ＤＯＳ１６ビット版とＭＳ－ＤＯＳ３２ビット版が用意されています。

１６ビット版と３２ビット版については、各ライブラリの仕様、インクルードファイルは、共通となっています。

ＭＳ－ＤＯＳ１６ビット版　ホストＣ－ＮＥＴライブラリ

動作環境

ホスト　　：ＮＥＣ ＰＣ－９８０１シリーズ
　　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ　　：ＭＳ－ＤＯＳ
コンパイラ　：Ｔｕｒｂｏ Ｃ++ Ｖｅｒ．１．０
　　Ｍｉｃｒｏｓｏｆｔ Ｃ Ｖｅｒ．６．０
　　Ｗａｔｃｏｍ Ｃ／Ｃ++ Ｖｅｒ．１１．０Ｊ

構成

＜ＰＣ－９８用＞

\lib\ltcnet.lib ･･･････Turbo C++ Ver. 1.0 用ライブラリ
\lib\lmcnet.lib ･･･････Microsoft C Ver. 6.0 用ライブラリ
\lib\lwcnet.lib ･･･････Watcom C/C++ Ver.11.0J 用ライブラリ
\include\cnet98.h ･････「基本ライブラリ」、「応用ライブラリ」用インクルードファイル
include\cnet98if.h ････「応用ライブラリ」用インクルードファイル

＜ＰＣ／ＡＴ用＞

\lib\ltcnetat.lib ･････Turbo C++ Ver. 1.0 用ライブラリ
\lib\lmcnetat.lib ･････Microsoft C Ver. 6.0 用ライブラリ
\lib\lwcnetat.lib ･････Watcom C/C++ Ver.11.0J 用ライブラリ
\include\cnetat.h ･････「基本ライブラリ」、「応用ライブラリ」用インクルードファイル
\include\cnetatif.h ･･･「応用ライブラリ」用インクルードファイル

注意

◎本ライブラリはラージモデル用です。
◎本ライブラリはプログラムのスタックを多く使用しています。
　スタックサイズを 4096 以上にしてください。
　Microsoft C の場合はリンカオプション/STACK でスタックサイズを 4096 以上にします。
　Turbo C++の場合はデフォルトのスタックサイズは 4096 です。
　4096 以上の指定する場合はグローバル変数_stklen で指定します。
　Watcom C/C++の場合はリンカオプション STACK でスタックサイズを 4096 以上にします。
　コンパイル、リンクの参考として**４．サンプルプログラム**の項目を参照してください。
◎Windows の MS-DOS プロンプトでの動作はサポートされません。

ＭＳ－ＤＯＳ３２ビット版　ホストＣ－ＮＥＴライブラリ

動作環境

ホスト　　：ＮＥＣ ＰＣ－９８０１シリーズ
　　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ　　：ＭＳ－ＤＯＳ
コンパイラ　：Ｗａｔｃｏｍ Ｃ／Ｃ++ Ｖｅｒ．１１．０Ｊ

構成

＜ＰＣ－９８用＞

\lib\xwcnet.lib ･･･････ Watcom C/C++ Ver.11.0J 用ライブラリ
\include\cnet98.h ･････「基本ライブラリ」、「応用ライブラリ」用インクルードファイル
\include\cnet98if.h ･･･「応用ライブラリ」用インクルードファイル

＜ＰＣ／ＡＴ用＞

\lib\xwcnetat.lib ･････ Watcom C/C++ Ver.11.0J 用ライブラリ
\include\cnetat.h ･････「基本ライブラリ」、「応用ライブラリ」用インクルードファイル
\include\cnetatif.h ･･･「応用ライブラリ」用インクルードファイル

注意

◎本ライブラリはプログラムのスタックを多く使用しています。
　スタックサイズを 4096 以上にしてください。
　リンカオプション STACK でスタックサイズを 4096 以上にします。
　コンパイル、リンクの参考として**４．サンプルプログラム**の項目を参照してください。
◎Windows の MS-DOS プロンプトでの動作はサポートされません。

２．基本ライブラリ

「基本ライブラリ」にはＣ－ＮＥＴを使用するために最低限必要なライブラリについて、記載されています。
各ライブラリ関数に対しては、詳細な解説、形式、インクルードファイル、プロトタイプ、引き数、戻り値が説明されています。

①Ｃ－ＮＥＴインターフェイスの初期化（C_NET_Init）
②Ｃ－ＮＥＴインターフェイスの終了処理（C_NET_Finish）
③受信割り込み処理関数の登録（C_NET_IRQ_Set）
④受信割り込み処理関数の登録解除（C_NET_IRQ_Clear）
⑤Ｃ－ＮＥＴ上のノード・マップの取得（C_NET_Get_Node_Map）
⑥受信ステータスの取得（C_NET_Receive_Status）
⑦受信処理（C_NET_Receive）
⑧送信ステータスの取得（C_NET_Send_Status）
⑨送信処理（C_NET_Send）

必要なインクルードファイルは
cnet98.h（PC-9801用）
cnetat.h（PC/AT用）
です。

C_NET_Init

機　能

Ｃ－ＮＥＴインターフェイスの初期化

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_Init(int node, unsigned int  module_base, int io_select,int  irq_select);
```

解　説

Ｃ－ＮＥＴインターフェイスを初期化するための制御関数です。Ｃ－ＮＥＴを使用する前には、必ず、この制御関数をコールする必要があります。（オープン関数に相当します。）引数の内、インターフェイス基板に関するものは、その基板に設定されている内容と同じ内容を指定してください。戻り値（６）は異常ではありませんので、リトライ可能です。

☆ＰＣ－９８０１用

① ***node*** は自局ノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***module_base*** はインターフェイス基板のバッファベースアドレスです。

値	定　数	意　味
0xC000	BUF_MEM_C000	0xC0000～C3FFFを指定します。
0xC400	BUF_MEM_C400	0xC4000～C7FFFを指定します。
0xC800	BUF_MEM_C800	0xC8000～CBFFFを指定します。
0xCC00	BUF_MEM_CC00	0xCC000～CFFFFを指定します。
0xD000	BUF_MEM_D000	0xD0000～D3FFFを指定します。
0xD400	BUF_MEM_D400	0xD4000～D7FFFを指定します。
0xD800	BUF_MEM_D800	0xD8000～DBFFFを指定します。
0xDC00	BUF_MEM_DC00	0xDC000～DFFFFを指定します。

③ ***io_select*** はインターフェイス基板のＩ／Ｏアドレスです。

値	定　数	意　味
０	IO_0	DxC0D0,C0D2,C0D6,C0D4,C0D8,C0DA,C0DC,C0DEを指定します。
１	IO_1	0xC1D0,C1D2,C1D4,C1D6,C1D8,C1DA,C1DC,C1DEを指定します。
２	IO_2	0xC2D0,C2D2,C2D4,C2D6,C2D8,C2DA,C2DC,C2DEを指定します。
３	IO_3	0xC3D0,C3D2,C3D4,C3D6,C3D8,C3DA,C3DC,C3DEを指定します。

④ ***irq_select*** はインターフェイス基板の割り込みレベルです。

値	定　数	意　味
0	INT_0	1R3 (INT0)を指定します。
1	INT_1	1R5 (INT1)を指定します。
2	INT_2	1R6 (INT2)を指定します。
3	INT_3	1R9 (INT3)を指定します。
4	INT_4	1R10(INT4)を指定します。
5	INT_5	1R12(INT5)を指定します。
6	INT_6	1R13(INT6)を指定します。

☆ＰＣ／ＡＴ用

① ***node*** は自局ノード番号です。

値	定　数	意　味
0	NODE_0	自局ノード番号０を指定します。
・	・	・
６３	NODE_63	自局ノード番号６３を指定します。

② ***module_base*** はインターフェイス基板のバッファベースアドレスです。

値	定　数	意　味
0xC800	BUF_MEM_C800	0xC8000～CBFFFを指定します。
0xCC00	BUF_MEM_CC00	0xCC000～CFFFFを指定します。
0xD000	BUF_MEM_D000	0xD0000～D3FFFを指定します。
0xD400	BUF_MEM_D400	0xD4000～D7FFFを指定します。
0xD800	BUF_MEM_D800	0xD8000～DBFFFを指定します。
0xDC00	BUF_MEM_DC00	0xDC000～DFFFFを指定します。
0xE000	BUF_MEM_E000	0xE0000～E3FFFを指定します。
0xE400	BUF_MEM_E400	0xE4000～E7FFFを指定します。

③ ***io_select*** はインターフェイス基板のＩ／Ｏアドレスです。

値	定　数	意　味
0	IO_0	アドレス＝0x0330～0337を指定します。
1	IO_1	アドレス＝0x0338～033Fを指定します。
2	IO_2	アドレス＝0x0340～0347を指定します。
3	IO_3	アドレス＝0x0348～034Fを指定します。

④ ***irq_select*** はインターフェイス基板の割り込みレベルです。

値	定　数	意　味
0	INT_0	IRQ3 を指定します。
1	INT_1	IRQ4 を指定します。
2	INT_2	IRQ5 を指定します。
3	INT_3	IRQ7 を指定します。
4	INT_4	IRQ9 を指定します。
5	INT_5	IRQ10を指定します。
6	INT_6	IRQ11を指定します。
7	INT_7	IRQ12を指定します。
8	INT_8	IRQ14を指定します。
9	INT_9	IRQ15を指定します。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了しました。
6	LINE_BUSY	伝送路が290ミリ秒以上ビジィ状態であったため中断しました。
－１	ARG_ERR	引数の値が異常です。
－３	ALREADY_EXIST	既に当該ノード番号がC-NET上に存在しています。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

例

Ｃ－ＮＥＴインターフェイスを初期化後、一文字”Ａ”を送信します。

```
#include <stdio.h>
#ifdef __TURBOC__
    #include <mem.h>
#elif defined __WATCOMC__
    #include <string.h>
#else
    #include <memory.h>
#endif
#include "cnet98.h"

static unsigned char message[ 128 ];
static unsigned char data[ 2048 ];

void main( void )
  {
    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        memset( message, 0x1a, 128 );     /* C-NETﾊﾞｯﾌｧの初期化 */
        memset( data, 0x1a, 2048 );    /* C-NETﾊﾞｯﾌｧの初期化 */

        *(short *)( &message[0]  ) = 0x0001;      /* ﾓｰﾄﾞ */
        *(short *)( &message[10] ) = 0x0001;      /* ﾃﾞｰﾀﾀｲﾌﾟ */
        *(long  *)( &message[12] ) = 1;           /* ﾃﾞｰﾀ有効長 */
        message[16]                = 0x01;        /* ﾀｲﾌﾟ */
        data[0] = 'A';
        if ( NO_ERROR == C_NET_Send( 8, 16, message, data ) )
          {
            while ( TRANSMIT == C_NET_Send_Status() )
              {

              }
          }
        C_NET_Finish();
      }
  }
```

留意事項　インターフェイス基板の設定と関数の引数が一致していない場合、本関数から戻らない（関数内でループする）場合があります。
その場合は、インターフェイス基板の設定と引数をチェックしてください。

C_NET_Finish

機　能

Ｃ－ＮＥＴインターフェイスの終了処理

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_Finish(void);
```

解　説

Ｃ－ＮＥＴインターフェイスによる通信を終了するための制御関数です。
Ｃ－ＮＥＴの使用を、終了する場合には、必ず、この制御関数をコールする必要があります。（クローズ関数に相当します。）この関数をコールしないでプログラムの実行を終了した場合、該当する割り込みベクタが復帰されず、該当する割り込みがマスクされません。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。

例

C_NET_Initを参照してください。

留意事項

ありません。

C_NET_IRQ_Set

機　能

受信割り込み処理関数の登録

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_IRQ_Set(void (*service_proc)());
```

解　説

受信処理をＣ－ＮＥＴインターフェイスからの受信割り込みで実行するような場合、この制御関数によって、あらかじめ処理関数を登録しておく必要があります。
登録はＣ－ＮＥＴインターフェイスの初期化後に行ってください。
受信処理関数内で使用可能なＣ－ＮＥＴライブラリは、C_NET_Receive関数のみです。
また、割り込まれる側のルーチンでは以下のＣ－ＮＥＴライブラリは使用できません。

C_NET_Receive_Status
C_NET_Receive
cupload
checkccmdack
checkcprmpt

① ***service_proc*** は割り込み処理関数のポインタです。
初期化の際にはＮＵＬＬになっています。
ＮＵＬＬの場合は何もしません。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－2	NO_CONNECT	まだ接続されていません。

例

割り込みで受信処理を行い、受信データをバッファに格納します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

static unsigned char message[128];
static unsigned char data[2048];
static char buff[ 100 ];        /* 受信バッファ */
static short buffp;             /* 受信バッファ インデックス */

void rcv_int( void );

void main( void )
  {
    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        buffp = 0;                      /* 受信バッファ インデックスのクリア */
        C_NET_IRQ_Set( rcv_int );       /* 割り込み処理関数の登録 */

        /* 何らかの処理が入ります。*/

        C_NET_IRQ_Clear();              /* 割り込み処理関数の登録解除 */
        C_NET_Finish();                 /* Ｃ－ＮＥＴの終了 */
      }
  }

/* Ｃ－ＮＥＴ受信割り込み処理*/
void rcv_int( void )
  {
    int node;/* 送信元ノード番号 */
    int len;/* 受信データ長 */
    short rcv_len;/* 受信有効長 */
    short i;

    if ( NO_ERROR == C_NET_Receive( message, data, &node, &len ) )
      {
        rcv_len = ( short )*(long *)&message[12];/* 受信有効長 */
        /* 受信データをバッファに格納 */
        for ( i = 0; i < rcv_len; i++ )
          {
            buff[ buffp++ ] = data[ i ];
            if ( buffp == 100 )
            buffp = 0;
          }
      }
  }
```

留意事項

後述する受信バッファ処理関数は、本関数を使用しＣ－ＮＥＴライブラリ内の割り込み処理関数を登録しています。
割り込みでは使用できるＣライブラリ関数に制限があります。
まず、ＤＯＳファンクションを呼び出している関数はすべて使用できません。
また、リエントラント（再入可能）でない関数も使用できません。

C_NET_IRQ_Clear

機　能

受信割り込み処理関数の登録解除

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_IRQ_Clear(void);
```

解　説

受信処理をＣ－ＮＥＴインターフェイスからの受信割り込みで実行しているような場合、この制御関数によって、その動作を停止できます。
再起動する場合は、再度、処理関数の登録を行う必要があります。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。

例

C_NET_IRQ_Setを参照してください。

留意事項

ありません。

C_NET_Get_Node_Map

機　能

Ｃ－ＮＥＴ上のノード・マップの取得

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
int C_NET_Get_Node_Map(unsigned char *node_map);
```

解　説

Ｃ－ＮＥＴ上に存在するノードを調べて報告します。
戻り値（６）は異常ではありませんので、リトライ可能です。

① ***node_map*** は、ノード・マップ格納用配列のポインタです。
配列は、unsigned char型で６４要素確保してください。
例）unsigned char node_map[64];
ノード番号をインデックスとして、配列内容が0xFFの場合は該当ノードが存在し、0x00の場合は存在しません。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
6	LINE_BUSY	伝送路が290ミリ秒以上ビジィ状態であったため中断しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

例

ノードマップを取得し、存在するノードを表示します。

```
#include <stdio.h>
#include "cnet98.h"

void main( void )
  {
    unsigned char node_map[ 64 ];
    short i;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        C_NET_Get_Node_Map( node_map );
        for ( i = 0; i < 64; i++ )
          {
            if ( node_map[ i ] == 0xff )
#ifdef __WATCOMC__
              cprintf( "%d \n", i );
#else
              printf( "%d \n", i );
#endif
          }
        C_NET_Finish();
      }
  }
```

留意事項

ありません。

C_NET_Receive_Status

機　能

受信ステータスの取得

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
int C_NET_Receive_Status(void);
```

解　説

受信の状態を調べて報告します。

戻り値

値	定　数	意　味
0	COMPLETE	受信完了しました。受信処理を行う必要があります。
3	READY	受信待機中です。
－2	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	ｲﾝﾀｰﾌｪｲｽまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	ｲﾝﾀｰﾌｪｲｽまたは制御関数に未知の異常が発生しました。

例

受信完了後、受信処理を行います。

```
#include <stdio.h>
#include "cnet98.h"

static unsigned char message[ 128 ];
static unsigned char data[ 2048 ];

void main( void )
  {
    int node;
    int length;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        while ( COMPLETE != C_NET_Receive_Status )
          {

          }
        C_NET_Receive( message, data, &node, &length );

        C_NET_Finish();
      }
  }
```

留意事項

ありません。

C_NET_Receive

機　能

受信処理

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_Receive(unsigned char *message,
                                  unsigned char *data, int *node, int *length);
```

解　説

受信したデータパケットのメッセージフィールドとデータフィールドを引き取るための制御関数です。

① ***message*** は受信メッセージフィールドを格納する配列のポインタです。
配列は、unsigned char型で１２８要素必要です。
例）unsigned char message[128];

② ***data*** は受信データフィールドを格納する配列のポインタです。
配列は、unsigned char型で２０４８要素必要です。
例）unsigned char data[2048];

③ ***node*** は送信先ノード番号を格納する変数のポインタです。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

④ ***length*** は受信データ長を格納する変数のポインタです。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
4	RX_NOTHING	まだ受信していません。受信待機中です。
－２	NO_CONNECT	まだ接続されていません。

例

C_NET_Receive_Statusを参照してください。

留意事項

メッセージフィールドとデータフィールドの構造については、**５．データパケット構造**を参照してください。

C_NET_Send_Status

機　能

送信ステータスの取得

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_Send_Status(void);
```

解　説

送信の状態を調べて報告します。
戻り値（１，２，５，６）は異常ではありませんので、リトライ可能です。

戻り値

値	定　数	意　味
０	COMPLETE	送信完了しました。
１	NOT_READY	送信先のノードが受信セットアップされていませんでした。
２	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
５	TRANSMIT	送信中です。
６	LINE_BUSY	伝走路が290ミリ秒以上ビジィ状態であったため中断しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

例

一文字 " A " を送信します。

```
#include <stdio.h>
#ifdef __TURBOC__
    #include <mem.h>
#elif defined __WATCOMC__
    #include <string.h>
#else
    #include <memory.h>
#endif
#include "cnet98.h"

static unsigned char message[ 128 ];
static unsigned char data[ 2048 ];

void main( void )
  {
    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        memset( message, 0x1a, 128 );     /* C-NETﾊﾞｯﾌｧの初期化 */
        memset( data, 0x1a, 2048 );       /* C-NETﾊﾞｯﾌｧの初期化 */

        *(short *)( &message[0]  ) = 0x0001;      /* ﾓｰﾄﾞ */
        *(short *)( &message[10] ) = 0x0001;      /* ﾃﾞｰﾀﾀｲﾌﾟ */
        *(long  *)( &message[12] ) = 1;           /* ﾃﾞｰﾀ有効長 */
        message[16]                = 0x01;        /* ﾀｲﾌﾟ */
        data[0] = 'A';
        if ( NO_ERROR == C_NET_Send( 8, 16, message, data ) )
          {
            while ( TRANSMIT == C_NET_Send_Status() )
              {

              }
          }
        C_NET_Finish();
      }
  }
```

留意事項

ありません。

C_NET_Send

機　能

送信処理

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
int C_NET_Send(int node, int length, unsigned char *message,unsigned char *data);
```

解　説

メッセージとデータをパケットにして送信するための制御関数です。

① ***node*** 送信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***length*** は送信データ長（１６～２０４８）です。
送信データ長の最小値は１６（バイト）です。
最小値に満たない場合も１６で指定します。

③ ****message*** は送信メッセージフィールド配列のポインタです。
配列はunsigned char型で１２８要素必要です。
例）unsigned char message[128];

④ ****data*** は送信データフィールド配列のポインタです。
配列はunsigned char型で２０４８要素必要です。
例）unsigned char data[2048];

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。

例

C_NET_Send_Statusを参照してください。

留意事項

メッセージフィールドとデータフィールドの構造については、**５．データパケット構造**を参照してください。

３．応用ライブラリ

「応用ライブラリ」には「基本ライブラリ」を使用した、よりユーザに近いレベルのライブラリについて記載されています。
各ライブラリ関数に対しては、詳細な解説、形式、インクルードファイル、プロトタイプ、引き数、戻り値が説明されています。

①バイナリーファイルの送信（cload）
②バイナリーファイルの受信（cupload）
③システムコマンドの発行（outccmd）
④コマンド応答チェック（checkccmdack）
⑤プロンプトチェック（checkcprmpt）
⑥データの送信（puts_cnet）
⑦データの受信（gets_cnet）
⑧受信バッファ処理の開始（cnetbuffopen）
⑨受信バッファ処理の終了（cnetbuffclose）
⑩送信元ノード番号、受信バッファの登録（setcnetrcv）
⑪送信元ノード番号、受信バッファの登録解除（clrcnetrcv）
⑫登録状況の参照（getcnetrcv）
⑬受信バッファ内有効データ長の取得（cnetbuffcheck）
⑭受信バッファから１バイトデータの読み込み（cnetbuffrcv）
⑮受信バッファの初期化（cnetbuffclr）
⑯エラーステータスの取得（cnetbuffstatus）

必要なインクルードファイルは
cnet98.h,cnet98if.h（PC-9801用）
cnetat.h,cnetatif.h（PC/AT用）
です。

cload

機　能

バイナリーファイルの送信

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short cload(short mode, short node, FILE *fp, char *ver,
                        void (*func)(long totalsize, long size), short rtime);
```

解　説

バイナリーファイル（システムファイル、ユーザファイル）を送信します。

① ***mode*** はファイルモードです。

値	定　数	意　味
0	SYSTEMFILE	システムファイルの送信を指定します。
1	USERFILE	ユーザファイルの送信を指定します。

② ***node*** 送信先のノード番号を指定します。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

③ ****fp*** は送信するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"r","b"です。

④ ****ver*** はシステムバージョンの文字列ポインタです。
システムファイルの転送時使用します。
ユーザファイル時はＮＵＬＬにしてください。

⑤ ****func*** はユーザ定義の関数のポインタです。
送信中に処理したいこと､送信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
送信データの総数（最初の呼出時のみ有効、以降は０）

size
今回送信した数を渡します。

この関数は送信の合間に（約２０４８バイト）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

⑥ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了しました。
－2	NO_CONNECT	まだ接続されていません。
－4	NOT_COMPLETE	まだ送信が完了していません。
－5	SEND_TO_ME	送信先が自局になっています。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
2	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
6	LINE_BUSY	伝走路が290ミリ秒以上ビジィ状態であったため中断しました。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
１０	FILE_SIZE_ERROR	ファイルのサイズが短いです。
１１	FILE_READ_ERROR	ファイル読み込みエラーです。
１３	TIME_OUT	送信タイムアウトエラー

NO_CONNECT,NOT_READY,NO_ACK,LINE_BUSYはリトライ後に返します。

例

バイナリーファイル"test.bin"を送信します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        if ( ( fp = fopen( "test.bin", "rb" ) ) != NULL )
          {
            cload( USERFILE, 8, fp, NULL, NULL, 5 );
          }
        C_NET_Finish();
      }
  }
```

留意事項

ユーザプログラムでファイルを受信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

cupload

機　能

バイナリーファイルの受信

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short cupload(short node, FILE *fp, void (*func)
                              (long totalsize,long size),short rtime);
```

解　説

バイナリーファイルを受信します。

① ***node*** は受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ****fp*** は受信したデータを保存するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"w","b"です。

③ ****func*** はユーザ定義の関数のポインタです。
受信中に処理したいこと､受信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
　受信データの総数（最初の呼出時のみ有効、以降は０）
size
　今回受信した数を渡します。

この関数は受信の合間に（約２０４８バイト）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

④ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
１３	TIME_OUT	受信待機中、一定時間受信ありません。
１２	FILE_WRITE_ERROR	ファイル書き込みエラーです。

NO_CONNECT,TIME_OUTはリトライ後に返します。

例

バイナリーファイルを受信しファイル"test.bin"に保存します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        if ( ( fp = fopen( "test.bin", "wb" ) ) != NULL )
          {
            cupload( 8, fp, NULL, 5 );
          }
        C_NET_Finish();
      }
  }
```

留意事項

ユーザプログラムでファイルを送信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

outccmd

機　能

システムコマンドの発行

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short outccmd(short node, char *s, short rtime);
```

解　説

システムコマンドを発行します。

① ***node*** は発行先のノード番号です

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***s*** はコマンド列のポインタです。ラインターミネータは内部で自動的に付加します。

③ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
2	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
6	LINE_BUSY	伝送路が290ミリ秒以上ビジィ状態であったため中断しました。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
１４	CMD_SIZE_ERROR	コマンドの長さが2048を越えています。

NO_CONNECT,NOT_READY,NO_ACK,LINE_BUSYはリトライ後に返します。

例

ファイル操作モードに移行します。

```
#include <stdio.h>
#include <string.h>
#include "cnet98.h"
#include "cnet98if.h"

#define RETRY_TIME     5
#define NODE_901       8

void main( void )
  {
    char cmdbuff[10];

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        strcpy(cmdbuff, "F");/* ﾌｧｲﾙ操作ﾓｰﾄﾞに移行 */
        if ( NO_ERROR == outccmd( NODE_901, cmdbuff, RETRY_TIME ) )
          {
            if ( NO_ERROR == checkccmdack( NODE_901, RETRY_TIME ) )
              {
                if ( NO_ERROR == checkcprmpt( NODE_901, RETRY_TIME ) )
#ifdef __WATCOMC__
                  cprintf( "File Mode \n" );
#else
                  printf( "File Mode \n" );
#endif
              }
          }
        C_NET_Finish();
      }
  }
```

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkccmdack

機　能

コマンド応答チェック

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short checkccmdack(short node, short rtime);
```

解　説

システムコマンドの応答をチェックしCMD_ACK('!'), CMD_NAK('%'), CMD_NON('?')が返されるまで指定時間待ちます。

CMD__ＡＣＫ ..... コマンド処理結果が正常
CMD__ＮＡＫ ..... コマンド処理結果が異常
CMD__ＮＯＮ ..... 未定義コマンド

① ***node*** は送受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
１５	CMD_ERROR	CMD_NAK, CMD_NONが返されました。（PC-9801用のみ）
３７	CMD_NAK	CMD_NAKが返されました。（PC/AT用のみ）
６３	CMD_NON	CMD_NONが返されました。（PC/AT用のみ）

TIME_OUT,NO_CONNECTはリトライ後返します。

例

outccmdを参照してください。

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkcprmpt

機　能

プロンプトチェック

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
short checkcprmpt(short node, short rtime);
```

解　説

システムプロンプト('>')が返されるまで指定時間待ちます。

① ***node*** 送受信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。

TIME_OUT,NO_CONNECTはのリトライ後返します。

例

outccmdを参照してください。

留意事項

システムプロンプトについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

puts_cnet

機　能

データの送信

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short puts_cnet( unsigned char *data, short length,
                                    short type,short node, short rtime );
```

解　説

指定されたノード番号にデータを送信します。

① ***data*** は送信データが格納されているバッファのポインタです。

② ***length*** は送信データ長（１から２０４８）です。

③ ***type*** は送信データの型です。

値	定　数	意　味
0	TEXT_TYPE	アスキーです。
1	BINARY_TYPE	バイナリです。

④ ***node*** は送信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
2	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
6	LINE_BUSY	伝走路が290ミリ秒以上ビジィ状態であったため中断しました。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

NO_CONNECT,NOT_READY,NO_ACK,LINE_BUSYはリトライ後返します。

例

データ0x00,0x01,0x02,0x03,0x04を送信します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    unsigned char buff[10];
    short i;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
      {
        for ( i = 0; i < 5; i++ )
            buff[ i ] = i;
        puts_cnet( buff, i, BINARY_TYPE, NODE_8, 5 );

        C_NET_Finish();
      }
  }
```

留意事項

９０Ｘ 基本ライブラリのLib_puts_cnetと１部、引き数、戻り値、型が違います。注意してください。

gets_cnet

機　能

データの受信

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short gets_cnet( unsigned char *data, short *length,
                                      short *type, short *node, short rtime );
```

解　説

指定されたノード番号からデータを受信します。

① ***data*** は受信データを格納するバッファのポインタです。

② ***length*** は受信データ長（１から２０４８）です。

③ ***type*** は受信データの型です。

値	定　数	意　味
０	TEXT_TYPE	アスキーです。
１	BINARY_TYPE	バイナリです。

④ ***node*** は送信元のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

NO_CONNECT,TIME_OUTはリトライ後返します。

例

データを受信し、表示します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    unsigned char buff[2048];
    short len, type, node;
    short i;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
      {
        gets_cnet( buff, &len, &type, &node, 5 );
        printf( "長さ:%d  型:%d  ﾉｰﾄﾞ:%d \n", len, type, node );
        for ( i = 0; i < len; i++ )
#ifdef __WATCOMC__
           cprintf( "%d \n", buff[ i ] );
#else
           printf( "%d \n", buff[ i ] );
#endif

        C_NET_Finish();
      }
    }
```

留意事項

９０Ｘ 基本ライブラリのLib_gets_cnetと１部、引き数、戻り値、型が違います。注意してください。

cnetbuffopen

機　能

受信バッファ処理の開始

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
void cnetbuffopen( void );
```

解　説

受信バッファ処理は、割り込みで受信処理を行い指定した受信バッファに受信データを格納します。
受信バッファ処理を行う場合は、Ｃ－ＮＥＴインターフェイスを初期化後、この関数をコールする必要があります。
受信バッファ処理に関連する関数は、

cnetbuffopen
cnetbuffclose
setcnetrcv
clrcnetrcv
getcnetrcv
cnetbuffcheck
cnetbuffrcv
cnetbuffclr
cnetbuffstatus

です。これらの関数は、受信バッファ処理を行う場合に使用できます。

また、次の関数は受信バッファ処理中は使用できません。

C_NET_IRQ_Set
C_NET_IRQ_Clear
C_NET_Receive_Status
C_NET_Receive
cupload
checkccmdack
checkcprmpt

戻り値

ありません。

例

４．サンプルプログラムを参照してください。

留意事項

ありません。

cnetbuffclose

機　能

受信バッファ処理の終了

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
void cnetbuffclose( void );
```

解　説

受信バッファ処理を終了する場合は、必ずこの関数をコールしてください。

戻り値

ありません。

例

４．サンプルプログラムを参照してください。

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

setcnetrcv

機　能

送信元ノード番号、受信バッファの登録

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
void setcnetrcv( short node, char *buff, short size );
```

解　説

送信元のノード番号と、ノード番号に対応する受信バッファのアドレスと大きさを登録します。
割り込み処理で、登録したノード番号に対応する受信バッファに、受信データを格納します。
登録していないノードからの受信データは捨てます。
送信元が複数（複数台の９０Ｘが接続されている）の場合は、この関数を必要な回数コールして登録を行います。
この関数は、受信バッファ処理の開始(cnetbuffopen)後に必ずコールしてください。

① ***node*** は送信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ****buff*** は受信バッファのポインタです。

③ ***size*** は受信バッファの大きさです。
６４バイト～１６３８４バイト

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。

例

４．サンプルプログラムを参照してください。

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

clrcnetrcv

機　能

送信元ノード番号、受信バッファの登録解除

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
void clrcnetrcv( short node );
```

解　説

送信元のノード番号と受信バッファの登録を解除します。

① ***node*** は送信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。
0xFF	NODE_ALL	0xFFを指定するとすべての登録を解除します。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。

例

ノード番号８の登録を解除します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    char buff[ 256 ];

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        cnetbuffopen();                    /* 受信ﾊﾞｯﾌｧ処理の開始 */
        setcnetrcv( NODE_8, buff, 256 );   /* 受信ﾊﾞｯﾌｧの登録 */

        /* 何らかの処理が入ります */

        clrcnetrcv( NODE_8 );              /* ﾉｰﾄﾞ番号8の登録解除 */

        /* 何らかの処理が入ります */

        cnetbuffclose();                   /* 受信ﾊﾞｯﾌｧ処理の終了 */
        C_NET_Finish();
      }
  }
```

留意事項　本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

getcnetrcv

機　能

登録状況の参照

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
void getcnetrcv( unsigned char *map );
```

解　説

現在登録されているノード番号を参照します。
登録とは、setcnetrcv関数による送信元ノード番号、受信バッファの登録を指します。

① ***map*** は登録ノード格納用配列のポインタです。
配列は、unsigned char型で６４要素確保してください。
例）unsigned char map[64];
ノード番号をインデックスとして、配列内容が0xFFの場合は該当ノードが登録済みで、0x00の場合は登録されていません。

戻り値

ありません。

例

登録状況を参照し、登録済のノード番号を表示します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    char buff[ 256 ];
    char unsigned char map[ 64 ];
    short i;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        cnetbuffopen();                     /* 受信ﾊﾞｯﾌｧ処理の開始 */
        setcnetrcv( NODE_8, buff, 256 );    /* 受信ﾊﾞｯﾌｧの登録 */

        /* 何らかの処理が入ります */

        getcnetrcv( map );                  /* 登録状況参照 */
        for ( i = 0; i < 64; i++ )
          {
            if ( map[ i ] == 0xff )
#ifdef __WATCOMC__
              cprintf( "%d \n", i );
#else
              printf( "%d \n", i );
#endif
          }

        cnetbuffclose();                    /* 受信ﾊﾞｯﾌｧ処理の終了 */
        C_NET_Finish();
      }
  }
```

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

cnetbuffcheck

機　能

受信バッファ内有効データ長の取得

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
short cnetbuffcheck( short node );
```

解　説

受信バッファ内の有効データ長を取得します。

① ***node***は受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
－１以外		有効データ長（バイト）を返します。
－１	ARG_ERR	引数の値が異常です。

例

４．サンプルプログラムを参照してください。

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

cnetbuffrcv

機　能

受信バッファから１バイトデータの読み込み

形　式

```
#include "cnet98.h"      /* PC/AT用は cnetat.h */
#include "cnet98if.h"    /* PC/AT用は cnetatif.h */
short cnetbuffrcv( short node );
```

解　説

受信バッファから読み込んだ１バイトのデータを符号拡張せずに、shortに変換してから返します。何らかのエラーが発生している場合は－１を返します。
エラーの内容はcnetbuffstatus関数で参照できます。

① ***node*** は受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
－１以外		受信データを返します。
－１		何らかのエラーが発生しています。

例

４．サンプルプログラムを参照してください。

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

cnetbuffclr

機　能

受信バッファの初期化

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
void cnetbuffclr( short node );
```

解　説

受信バッファを初期化します。
すでに受信しているデータは無くなります。

① ***node*** は受信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。
0xFF	NODE_ALL	0xFFを指定するとすべての受信バッファを初期化します。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。

例

エラーが発生している場合、エラーコードを表示し受信バッファを初期化します。

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

void main( void )
  {
    char buff[ 256 ];
    short ret;

    if ( NO_ERROR == C_NET_Init( NODE_0, BUF_MEM_C000, IO_1, INT_5 ) )
      {
        cnetbuffopen();                     /* 受信バッファ処理の開始 */
        setcnetrcv( NODE_8, buff, 256 );    /* 受信バッファの登録 */

        /* 何らかの処理が入ります */

        ret = cnetbuffstatus( NODE_8 );     /* エラーステータスの取得 */
        if ( NO_ERROR != ret )
          {
#ifdef __WATCOMC__
            cprintf( "%d \n", ret );
#else
            printf( "%d \n", ret );
#endif
            cnetbuffclr( NODE_8 );          /* 受信バッファのクリア */
          }

        cnetbuffclose();                    /* 受信バッファ処理の終了 */
        C_NET_Finish();
      }
  }
```

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

cnetbuffstatus

機　能

エラーステータスの取得

形　式

```
#include "cnet98.h"       /* PC/AT用は cnetat.h */
#include "cnet98if.h"     /* PC/AT用は cnetatif.h */
short cnetbufferr( short node );
```

解　説

エラーが発生しているか調べます。
エラーが発生すると、ノードごとにエラー標識がセットされます。
エラー標識はこの関数を呼び出すまでセットされたままで、呼び出すことによりクリアされます。

① ***node*** は送受信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
２０	NOT_REGIST	ノードが登録されていません。
２１	BUFF_OVER	受信バッファがオーバフローしました。
２２	BUFF_DATA_NOTHING	受信バッファにデータがありません。

例

cnetbuffclrを参照してください。

留意事項

本関数は受信バッファ処理関数です。
受信バッファ処理については、cnetbuffopenの解説を参照してください。

４．サンプルプログラム

4.1　MS-DOS16ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ (Turbo C++/Microsoft C編)

ＭＳ－ＤＯＳ１６ビット版ホストＣ－ＮＥＴライブラリを使用したサンプルプログラムが

\sample\dload901.c ･･･････ サンプルプログラム
\sample\dload901.exe ･････ tc.batを使用し作成した実行ファイル
tc.bat ･･･････････････････ dload901.cのコンパイル、リンクを行うバッチファイル
(Turbo C++用)
msc.bat ･･･････････････････ dload901.cのコンパイル、リンクを行うバッチファイル
(Microsoft C用)

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

dload901 [OPTION [OPTION...]] filename

① ***-Mn*** ･･･････････････ ホストのノード番号です。ｎは０～７省略時は０です。

② ***-Tn*** ･･･････････････ ターゲットのノード番号です。ｎは８～６３省略時は８です。

③ ***-On*** ･･･････････････ インターフェースボードのＩ／Ｏアドレスを指定します。
◎ＰＣ－９８０１用
アドレスの上位２バイトを指定します。
ｎはC0, C1, C2, C3省略時はC1 です。
◎ＰＣ／ＡＴ用
ｎはC0(330), C1(338), C2(340), C3(348)省略時はC0(330) です。

④ ***-Bn*** ･･･････････････ インターフェースボードのバッファメモリアドレスの上位２バイトを指定します。
◎ＰＣ－９８０１用
ｎはC0, C4, C8, CC, D0, D4, D8, DC省略時はC4です。
◎ＰＣ／ＡＴ用
ｎはC8, CC, D0, D4, D8, DC, E0, E4省略時はC8です。

⑤ ***-In*** ･･･････････････ インターフェースボードの割り込みレベルを指定します。
◎ＰＣ－９８０１用
ｎは０～６省略時は５です。
◎ＰＣ／ＡＴ用
ｎは3, 4, 5, 7, 9, 10, 11, 12, 14, 15省略時は１０です。

⑥ ***filename*** ･･･････ ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

4.2　MS-DOS16 ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ(Wactom C/C++編)

ＭＳ－ＤＯＳ１６ビット版ホストＣ－ＮＥＴライブラリを使用したサンプルプログラムが

＜ＰＣ－９８用＞
\sample\lw_cnet9.c ･･･････サンプルプログラム
\sample\lw_cnet9.exe ･････wtcn98.bat を使用し作成した実行ファイル
wtcn98.bat ･･･････････････lw_cnet9.c のコンパイル､リンクを行うバッチファイル

＜ＰＣ／ＡＴ用＞
\sample\lw_cneta.c ･･･････サンプルプログラム
\sample\lw_cneta.exe ･････wtcnat.bat を使用し作成した実行ファイル
wtcnat.bat ･･･････････････lw_cneta.c のコンパイル､リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

lw_cnet9(lw_cneta) [OPTION [OPTION...]] filename

① ***-Mn*** ･･･････････････ホストのノード番号です。ｎは０～７省略時は０です。

② ***-Tn*** ･･･････････････ターゲットのノード番号です。ｎは８～６３省略時は８です。

③ ***-On*** ･･･････････････インターフェースボードのＩ／Ｏアドレスを指定します。
◎ＰＣ－９８０１用
ｎはC0, C1, C2, C3省略時はC1です。
◎ＰＣ／ＡＴ用
ｎはC0(330), C1(338), C2(340), C3(348)省略時はC0(330)です。

④ ***-Bn*** ･･･････････････インターフェースボードのバッファメモリアドレスの上位２バイトを指定します。
◎ＰＣ－９８０１用
ｎはC0, C4, C8, CC, D0, D4, D8, DC省略時はC4です。
◎ＰＣ／ＡＴ用
n はC8, CC, D0, D4, D8, DC, E0, E4省略時はC8です。

⑤ ***-In*** ･･･････････････インターフェースボードの割り込みレベルを指定します。
◎ＰＣ－９８０１用
ｎは０～６省略時は５です。
◎ＰＣ／ＡＴ用
ｎは3, 4, 5, 7, 9, 10, 11, 12, 14, 15省略時は10です。

⑥ ***filename*** ･･･････････ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcn98.bat(wtcnat.bat)を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-16-BIT
[Image Type] を Executable [.exe]
にしてください。
スタックサイズの指定、メモリモデルの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

4.3　ＭＳ－ＤＯＳ３２ビット用サンプルプログラム

ＭＳ－ＤＯＳ３２ビット版ホストＣ－ＮＥＴライブラリを使用したサンプルプログラムが

＜ＰＣ－９８用＞
\sample\xw_cnet9.c ・・・・・・・サンプルプログラム
\sample\xw_cnet9.exe ・・・・・wtcn9832.bat を使用し作成した実行ファイル
wtcn9832.bat ・・・・・・・・・・・・xw_cnet9.c のコンパイル、リンクを行うバッチファイル

＜ＰＣ／ＡＴ用＞
\sample\xw_cneta.c ・・・・・・・サンプルプログラム
\sample\xw_cneta.exe ・・・・・wtcnat32.bat を使用し作成した実行ファイル
wtcnat32.bat ・・・・・・・・・・・・xw_cneta.c のコンパイル、リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

xw_cnet9(xw_cneta) [OPTION [OPTION...]] filename

① ***-Mn*** ・・・・・・・・・・・・ホストのノード番号です。ｎは０～７省略時は０です。

② ***-Tn*** ・・・・・・・・・・・・・ターゲットのノード番号です。ｎは８～６３省略時は８です。

③ ***-On*** ・・・・・・・・・・・・インターフェースボードのＩ／Ｏアドレスを指定します。
◎ＰＣ－９８０１用
アドレスの上位２バイトを指定します。
ｎはC0, C1, C2, C3省略時はC1です。
◎ＰＣ／ＡＴ用
ｎはC0(330), C1(338), C2(340), C3(348)省略時はC0(330)です。

④ ***-Bn*** ・・・・・・・・・・・・インターフェースボードのバッファメモリアドレスの上位２バイトを指定します。
◎ＰＣ－９８０１用
ｎはC0, C4, C8, CC, D0, D4, D8, DC省略時はC4です。
◎ＰＣ／ＡＴ用
ｎはC8, CC, D0, D4, D8, DC, E0, E4省略時はC8です。

⑤ ***-In*** ・・・・・・・・・・・・・インターフェースボードの割り込みレベルを指定します。
◎ＰＣ－９８０１用
ｎは０～６省略時は５です。
◎ＰＣ／ＡＴ用
ｎは3, 4, 5, 7, 9, 10, 11, 12, 14, 15省略時は10です。

⑥ ***filename*** ・・・・・・・・ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcn9832.bat(wtcnat32.bat)を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-32-BIT
[Image Type] を DOS4GW Executable [.exe]
にしてください。
スタックサイズの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

4.4　受信バッファ処理を使用した、割り込みによる受信

２台の９０１（ノード８，９）からのデータを受信し、何らかの処理を行います。
このサンプルはＰＣ－９８０１用です。

例

```
#include <stdio.h>
#include "cnet98.h"
#include "cnet98if.h"

#define MY_NODE          0
#define NO1_901_NODE     8
#define NO2_901_NODE     9

void main( void )
  {
    int ret;
    char no1buff[ 256 ];    /* No1 901用受信ﾊﾞｯﾌｧ */
    char no2buff[ 256 ];    /* No2 901用受信ﾊﾞｯﾌｧ */
    short rcvcnt;           /* 有効ﾃﾞｰﾀ長 */
    short rcvdata;          /* 受信ﾃﾞｰﾀ */

    /* Ｃ－ＮＥＴの初期化 */
    ret = C_NET_Init( MY_NODE, BUF_MEM_C400, IO_1, INT_5 );
    if ( ret == NO_ERROR )
      {
        cnetbuffopen();/* 受信ﾊﾞｯﾌｧ処理の開始 */
        setcnetrcv( NO1_901_NODE, no1buff, 256 );   /* 受信ﾊﾞｯﾌｧの登録 */
        setcnetrcv( NO2_901_NODE, no2buff, 256 );   /*                  */
        /* 何らかの処理が入ります */
        rcvcnt = cnetbuffcheck( NO1_901_NODE );     /* ﾊﾞｯﾌｧをﾁｪｯｸ */
        if ( rcvcnt > 0 )
          {
            /* No1 901のﾃﾞｰﾀを読み込む */
            rcvdata = cnetbuffrcv( NO1_901_NODE );
            /* 何らかの処理がはいります */
          }
        rcvcnt = cnetbuffcheck( NO2_901_NODE );     /* ﾊﾞｯﾌｧをﾁｪｯｸ */
        if ( rcvcnt > 0 )
          {
            /* No2 901のﾃﾞｰﾀを読み込む */
            rcvdata = cnetbuffrcv( NO2_901_NODE );
            /* 何らかの処理がはいります。*/
          }
        /* 受信ﾊﾞｯﾌｧ処理､C-NETを終了する場合 */
        cnetbuffclose();      /* 受信ﾊﾞｯﾌｧ処理の終了 */
        C_NET_Finish();       /* C-NETの終了 */
      }
  }
```

５．データパケット構造

Ｃ－ＮＥＴで通信を行うためのデータパケット構造について説明します。
データパケットは「メッセージ・フィールド」と「データ・フィールド」で構成されます。

６．メッセージフィールド構造

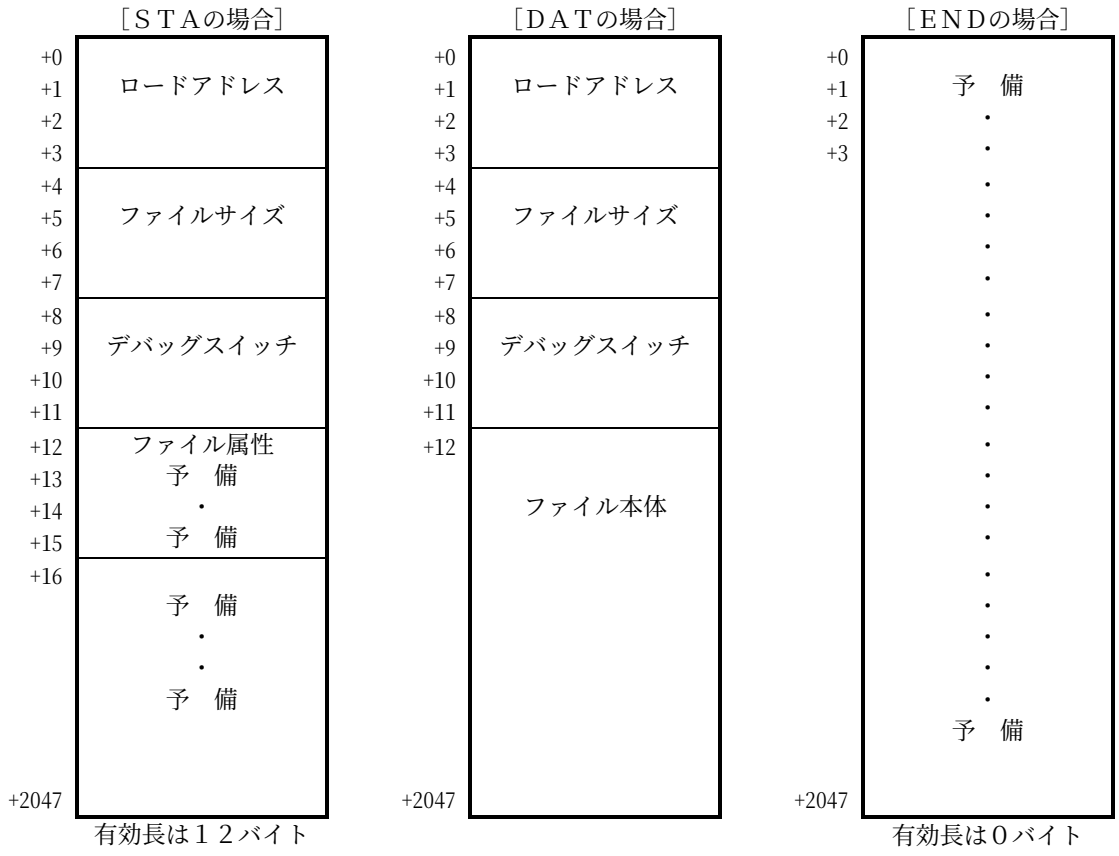
＋０ ＋１	モード	送信モードを指定します。 [0x0000]システムの転送 [0x0001]上記以外
＋２ ＋３	メジャーバージョン	システム転送時、システムのバージョンをASCIIで格納 例）1.23
＋４ ＋５	マイナーバージョン	’１’ ’ ’ ’２’ ’３’　空きはNULLで埋めます。 ＋２ ＋３ ＋４ ＋５
＋６ ＋７	予　　備	システム転送以外は0x1aで埋めます。
＋８ ＋９	予　　備	
＋１０ ＋１１	データタイプ	[0x000]ファイル転送時 [0x001]ファイル転送以外
＋１２ ＋１３ ＋１４ ＋１５	データ有効長	データフィールドの有効長
＋１６	タイプ	データフィールドの内容 [0x00]ASCII [0x01]バイナリー
＋１７ ・ ・ ・ ＋２７	予　　備 ・ ・ ・ 予　　備	
＋２８ ＋２９ ＋３０ ・ ・ ・ ・ ・ ・ ・ ・ ・ ＋127		ファイル転送時、データフィールドの種別を指定します。 ’Ｓ’ ’Ｔ’ ’Ａ’ ’￥０’　ﾃﾞｰﾀﾊﾟｹｯﾄがヘッダの時 ＋２８ ＋２９ ＋３０ ＋３１　（ファイル転送の開始） ’Ｄ’ ’Ａ’ ’Ｔ’ ’￥０’　ﾃﾞｰﾀﾊﾟｹｯﾄがﾌｧｲﾙ本体時 ＋２８ ＋２９ ＋３０ ＋３１ ’Ｅ’ ’Ｎ’ ’Ｄ’ ’￥０’　ファイル転送の終了 ＋２８ ＋２９ ＋３０ ＋３１ ファイル転送以外は0x1aで埋めます。

予備、空きデータは0x1aで埋めます。

7．データフィールド構造

7.1　システムファイル転送

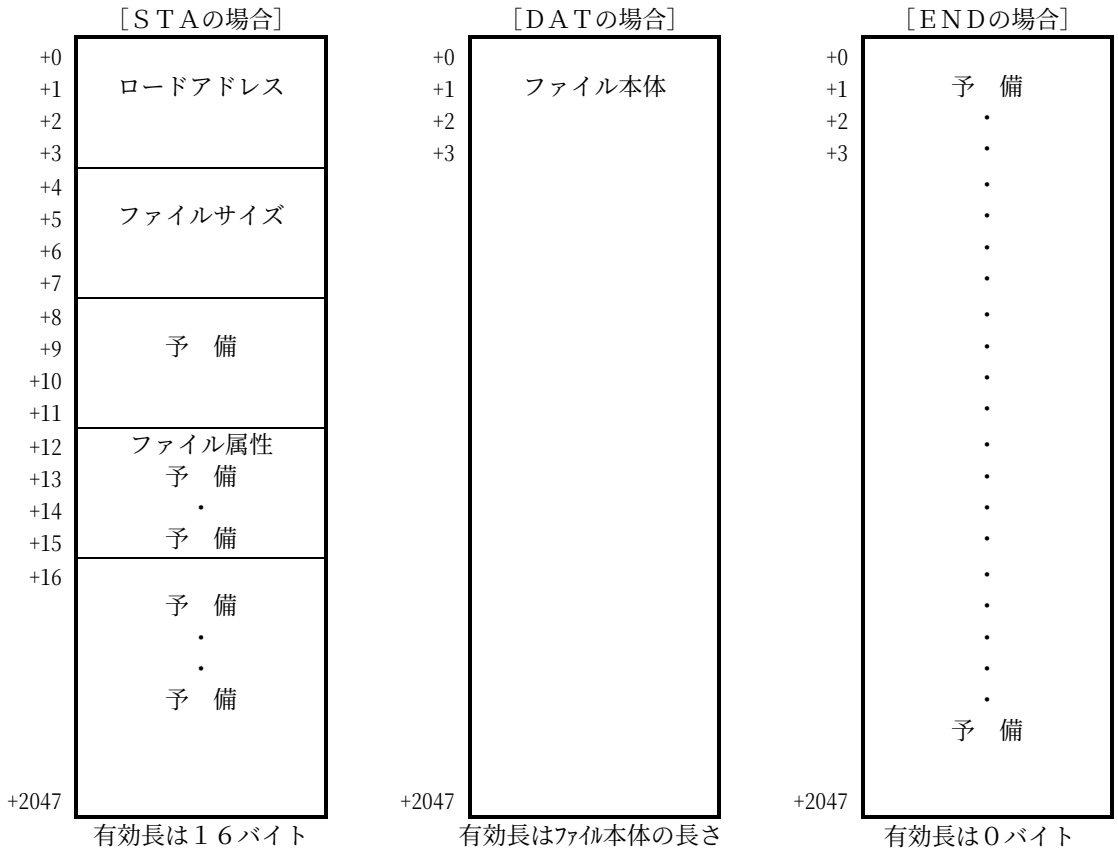
予備、空きデータは0x1aで埋めます。



○システムファイルの先頭１６バイトにはロードアドレス、ファイルサイズ、デバッグスイッチ、ファイル属性が含まれていて以後、システム本体となっています。
ＳＴＡの先頭１６バイトには、システムファイルの先頭１６バイトと同じものを格納します。
○ＤＡＴのロードアドレス、ロードサイズ、デバッグスイッチは先頭のＤＡＴにのみ含めます。
これらは、システムファイルの先頭１６バイトからファイル属性を除いたものです。
以後はファイル本体の続きになります。

7.2　ユーザファイル転送

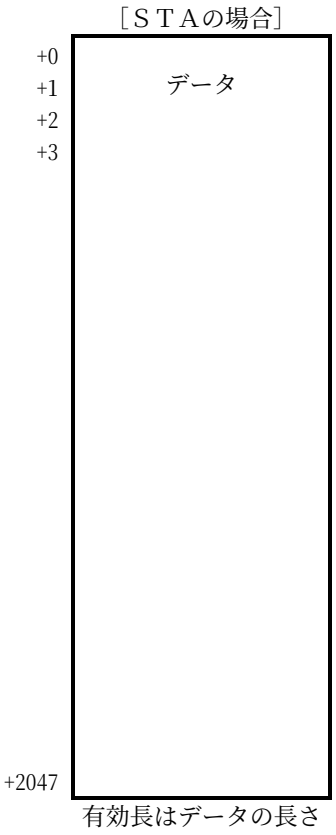
予備、空きデータは0x1aで埋めます。



○ユーザファイル先頭１６バイトには、ロードアドレス、ファイルサイズ、予備、ファイル属性が含まれ、以後ファイル本体となります。
ＳＴＡの先頭１６バイトには、ユーザーファイル先頭１６バイトと同じものを格納します。

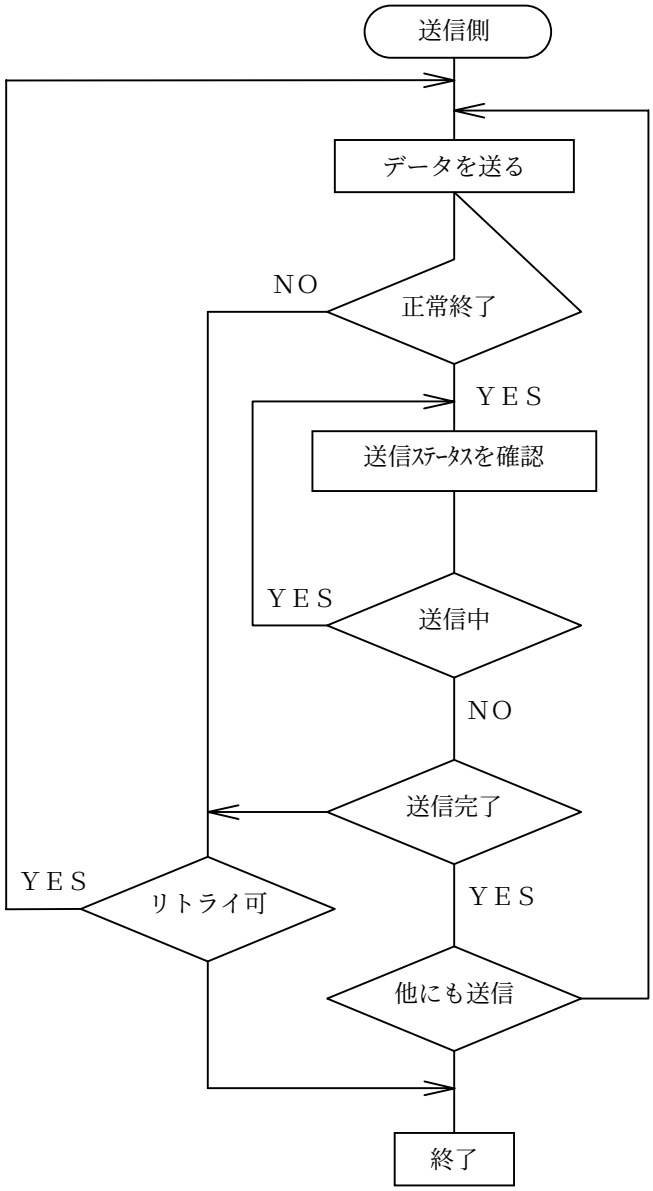
7.3　ファイル転送以外

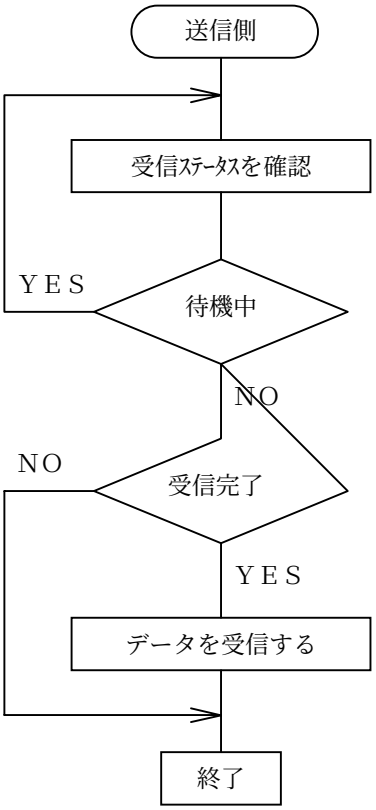
予備、空きデータは、0x1aで埋めます。



８．通信手順

ファイル転送、データの転送共、次の方法で通信できます。
ファイル転送時は最初にＳＴＡのパケットを送り、次にＤＡＴのパケットがなくなるまで送ります。
最後にＥＮＤのパケットを送りファイル転送の終了となります。





第２編

ＤＯＳ版
ホストＣ－ＬＩＮＫライブラリ

アルファベット順ライブラリ目次

項番	ライブラリ名	ページ
1	checkcmdack_clink	100
2	checkcmdack_rs232c	85
3	checkprmpt_clink	101
4	checkprmpt_rs232c	86
5	finish_clink	93
6	finish_rs232c	78
7	gets_clink	104
8	gets_rs232c	88
9	init_clink	92
10	init_rs232c	76
11	load_clink	94
12	load_rs232c	79
13	outcmd_clink	98
14	outcmd_rs232c	83
15	puts_clink	102
16	puts_rs232c	87
17	upload_clink	96
18	upload_rs232c	81

１．概　　要

本ライブラリは、ＣＳＣ９０Ｘとホスト間をＣ－ＬＩＮＫによる通信を行うライブラリです。
本ライブラリには、ＭＳ－ＤＯＳ１６ビット版とＭＳ－ＤＯＳ３２ビット版が用意されています。

１６ビット版と３２ビット版については、各ライブラリの仕様、インクルードファイルは、共通となっています。

ＭＳ－ＤＯＳ１６ビット版ホストＣ－ＬＩＮＫライブラリ

動作環境

ホスト　　：ＮＥＣ ＰＣ－９８０１シリーズ
　　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ　　：ＭＳ－ＤＯＳ
コンパイラ　：Ｔｕｒｂｏ Ｃ++ Ｖｅｒ．１．０
　　Ｍｉｃｒｏｓｏｆｔ Ｃ Ｖｅｒ．６．０
　　Ｗａｔｃｏｍ Ｃ／Ｃ++ Ｖｅｒ．１１．０Ｊ

構成

\lib\ltclink.lib･･･････Turbo C++ Ver. 1.0用ライブラリ
\lib\lmclink.lib･･･････Microsoft C Ver. 6.0用ライブラリ
\lib\lwclink.lib･･･････Watcom C/C++ Ver.11.0J用ライブラリ
\include\clink.h･･･････ＲＳ－２３２Ｃ，Ｃ－ＬＩＮＫライブラリ用インクルードファイル
\include\xsio.h･･･････clink.h にインクルードされます

注意

◎本ライブラリはラージモデル用です。
◎本ライブラリはプログラムのスタックを多く使用しています。
　スタックサイズを4096以上にしてください。
　Microsoft Cの場合はリンカオプション/STACKでスタックサイズを4096以上にします。
　Turbo C++の場合はデフォルトのスタックサイズは4096です。
　4096以上の指定する場合はグローバル変数_stklenで指定します。
　Watcom C/C++の場合はリンカオプションSTACKでスタックサイズを4096以上にします。
　コンパイル、リンクの参考として**４．サンプルプログラム**の項目を参照してください。
◎WindowsのMS-DOSプロンプトでの動作はサポートされません。

ＭＳ－ＤＯＳ３２ビット版ホストＣ－ＬＩＮＫライブラリ

動作環境

ホスト　　：ＮＥＣ ＰＣ－９８０１シリーズ
　　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ　　：ＭＳ－ＤＯＳ
コンパイラ　：Ｗａｔｃｏｍ Ｃ／Ｃ++ Ｖｅｒ．１１．０Ｊ

構成

\lib\xwclink.lib ・・・・・・Watcom C/C++ Ver.11.0J用ライブラリ
\include\clink.h ・・・・・・ＲＳ－２３２Ｃ，Ｃ－ＬＩＮＫライブラリ用インクルードファイル
\include\xsio.h ・・・・・・・clink.h にインクルードされます

注意

◎本ライブラリはプログラムのスタックを多く使用しています。
　スタックサイズを4096以上にしてください。
　リンカオプションSTACKでスタックサイズを4096以上にします。
　コンパイル、リンクの参考として**４．サンプルプログラム**の項目を参照してください。
◎WindowsのMS-DOSプロンプトでの動作はサポートされません。

２．ＲＳ２３２Ｃライブラリ

本ライブラリは、通常のＲＳ２３２Ｃで通信を行うためのものです。

init_rs232c

機　能

ＲＳ－２３２Ｃインターフェイスの初期化

形　式

```
#include "clink.h"
short init_rs232c( short machine, short speed, short length,
                   short parities, short stops, short xc, short channel );
```

解　説

ＲＳ－２３２Ｃインターフェイスを初期化します。（オープン関数に相当します。）
通常のＲＳ－２３２Ｃ通信とＣ－ＬＩＮＫ通信は同時に使用できますが、Ｃ－ＬＩＮＫの使用中（init_clinkから finish_clink まで）は、復帰（ＣＲ）／改行（ＬＦ）／タブ（ＴＡＢ）コードを除いて制御コードを使うような通常のＲＳ－２３２Ｃ通信は使用できません。
従いまして、以下のＲＳ－２３２Ｃ関数は、Ｃ－ＬＩＮＫ使用中は使用できません。

load_rs232c
upload_rs232c

その他の関数も上記制約を受けます。

① ***machine*** は使用するホストマシンです。

値	定　数	意　味
0	PC9801	ＰＣ９８シリーズおよび互換機
1	IBMPC	ＩＢＭＰＣおよび互換機
2	J3100	Ｊ３１００シリーズ
3	AX	ＡＸマシン

② ***speed*** は通信速度です。

値	定　数	意　味
0	B75	７５ｂｐｓ
1	B150	１５０ｂｐｓ
2	B300	３００ｂｐｓ
3	B600	６００ｂｐｓ
4	B1200	１２００ｂｐｓ
5	B2400	２４００ｂｐｓ
6	B4800	４８００ｂｐｓ
7	B9600	９６００ｂｐｓ
8	B19200	１９２００ｂｐｓ
9	B38400	３８４００ｂｐｓ

③ ***length*** はデータ長です。

値	定　数	意　味
0	L5	５ｂｉｔｓ
1	L6	６ｂｉｔｓ
2	L7	７ｂｉｔｓ
3	L8	８ｂｉｔｓ

④ ***parities*** はパリティです。

値	定数	意味
0	PNONE	パリティなし
1	PODD	奇数パリティ
3	PEVEN	偶数パリティ

⑤ ***stops*** はストップビット

値	定数	意味
0	S1	１ｂｉｔｓ
1	S15	１．５ｂｉｔｓ
3	S2	２ｂｉｔｓ

⑥ ***xc*** はＸフローコントロールです。

値	定数	意味
0	XCOFF	フロー制御なし
1	XCON	フロー制御を行う

⑦ ***channel*** はＲＳ－２３２Ｃのポートです。

値	定数	意味
0	FIRST	チャンネル０
1	SECOND	チャンネル１

戻り値

値	定数	意味
0	NO_ERROR	常に正常終了

例

load_rs232cを参照してください。

留意事項

個々のマシンの基本仕様にない設定はできません。

finish_rs232c

機　能

ＲＳ－２３２Ｃインターフェイスの終了処理

形　式

```
#include "clink.h"
void finish_rs232c( void );
```

解　説

ＲＳ－２３２Ｃインターフェイスを終了させます。
ＲＳ－２３２Ｃを終了する前には、必ず、この関数を呼び出す必要があります。
（クローズ関数に相当します。）

戻り値

ありません。

例

load_rs232cを参照してください。

留意事項

ありません。

load_rs232c

機　能

ファイルの送信

形　式

```
#include "clink.h"
short load_rs232c(short mode, FILE *fp,
                  void (*func)(long totalsize, long size), short rtime);
```

解　説

ファイル（システムファイル、ユーザーファイル）を送信します。

① ***mode*** で送信するファイルの種別を指定します。

値	定　数	意　味
0	SYSTEMFILE	システムファイルの送信を指定します。
1	USERFILE	ユーザファイルの送信を指定します。

② ****fp*** は送信するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"r","b"です。

③ ****func*** はユーザー定義の関数のポインタです。
送信中に処理したいこと、送信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
　送信データの総バイト数（最初の呼出時のみ有効、以降は０）

size
　今回送信したバイト数

を渡します。
この関数は送信の合間に（約２０４８バイト送信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

④ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　　数	意　　味
０	NO_ERROR	正常終了しました。
１０	FILE_SIZE_ERROR	ファイルのサイズが短すぎます。 （１６バイト以下）
１１	FILE_READ_ERROR	ファイル読み込みエラーです。
１３	TIME_OUT	タイムアウトエラーです。
２１	NAK	送信先が受信準備できていません。

例

ファイル"test.bin"を送信します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == init_rs232c( PC9801, B9600,L8, PNONE, S1, XCOFF, FIRST )
      {
        if ( ( fp = fopen( "test.bin", "rb" ) ) != NULL )
          {
            load_rs232c( USERFILE, fp, NULL, NULL, 5 );
          }
        finish_rs232c();
      }
  }
```

留意事項

ユーザプログラムでファイルを受信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

upload_rs232c

機　能

ファイルの受信

形　式

```
#include "clink.h"
short upload_rs232c(FILE *fp, void (*func)(long totalsize,long size),
                                                    short rtime);
```

解　説

ファイルを受信します。

① ***fp*** は受信したデータを保存するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"w","b"です。

② ***func*** はユーザー定義の関数のポインタです。
受信中に処理したいこと、受信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
　受信データの総数（最初の呼出時のみ有効、以降は０）

size
　今回受信した数

を渡します。
この関数は受信の合間に（約２０４８バイト受信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

③ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　　数	意　　味
０	NO_ERROR	正常終了しました。
１２	FILE_WRITE_ERROR	ファイル書き込みエラーです。
１３	TIME_OUT	受信待機中、一定時間受信ありません。

TIME_OUTはリトライ後に返します。

例

バイナリーファイルを受信しファイル"test.bin"に保存します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST)
      {
        if ( ( fp = fopen( "test.bin", "wb" ) ) != NULL )
          {
            upload_rs232c( fp, NULL, 5 );
          }
        finish_rs232c();
      }
  }
```

留意事項

ユーザプログラムでファイルを送信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

outcmd_rs232c

機　能

システムコマンドの発行

形　式

```
#include "clink.h"
short outcmd_rs232c(char *s);
```

解　説

システムコマンドを発行します。

① ***s*** はコマンド列のポインタで、ラインターミネータは内部で自動的に付加します。
コマンドの長さは最大１２８バイトです。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了です。
－1	ARG_ERR	引数の値が異常です。

例

ファイル操作モードに移行します。

```
#include <stdio.h>
#include <string.h>
#include "clink.h"

#define RETRY_TIME    5

void main( void )
  {
    char cmdbuff[ 10 ];

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST)
      {
        strcpy( cmdbuff, "F" );      /* ファイル操作モード移行コマンド */
        if ( NO_ERROR == outcmd_rs232c( cmdbuff ) )
          {
            if ( NO_ERROR == checkcmdack_rs232c( RETRY_TIME ) )
              {
                if ( NO_ERROR == checkprmpt_rs232c( RETRY_TIME ) )
#ifdef __WATCOMC__
                  cprintf( "ファイル操作モードに移行成功\n" );
#else
                  printf( "ファイル操作モードに移行成功\n" );
#endif
              }
          }
        finish_rs232c();
      }
  }
```

留意事項　システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkcmdack_rs232c

機　能

コマンド応答チェック

形　式

```
#include "clink.h"
short checkcmdack_rs232c(short rtime);
```

解　説

システムコマンドの応答をチェックしCMD_ACK('!'), CMD_NAK('%'), CMD_NON('?')が返されるまで指定時間待ちます。

ＣＭＤ＿ＡＣＫ‥‥‥‥‥‥ コマンド処理結果が正常です。
ＣＭＤ＿ＮＡＫ‥‥‥‥‥‥ コマンド処理結果が異常です。
ＣＭＤ＿ＮＯＮ‥‥‥‥‥‥ 未定義コマンドです。

① ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。 （CMD_ACKが返されました）
１３	TIME_OUT	受信待機中、一定時間受信がありません。
３７	CMD_NAK	CMD_NAKが返されました。
６３	CMD_NON	CMD_NONが返されました。

TIME_OUTはリトライ後返します。

例

outcmd_rs232cを参照してください。

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkprmpt_rs232c

機　能

プロンプトチェック

形　式

```
#include "clink.h"
short checkprmpt_rs232c(short rtime);
```

解　説

システムプロンプト('>')が返されるまで指定時間待ちます。

① ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。

TIME_OUTはリトライ後返します。

例

outcmd_rs232cを参照してください。

留意事項

システムプロンプトについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

puts_rs232c

機　能

データの送信

形　式

```
#include "clink.h"
short puts_rs232c( unsigned char *data, short length );
```

解　説

指定されたノード番号にデータを送信します。

① ***data*** は送信データが格納されているバッファのポインタです。

② ***length*** は送信データ長で、最大１２８バイトです。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了です。
－1	ARG_ERR	引数の値が異常です。

例

データ0x00,0x01,0x02,0x03,0x04を送信します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    unsigned char buff[ 130 ];
    short i;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST );
      {
        for ( i = 0; i < 5; i++ )
            buff[ i ] = i;
        puts_rs232c( buff, i );
        finish_rs232c();
      }
  }
```

留意事項

ありません。

gets_rs232c

機　能

データの受信

形　式

```
#include "clink.h"
short gets_rs232c( short option, short req_length,
                                    unsigned char *data, short *length, short rtime );
```

解　説

データを受信します。

① ***option*** は入力完了待ちオプションです。

値	定　数	意　味
0	NO_WAIT	受信バッファにデータがある場合は引き取ります。 なければ、そのままリターンします。
1	WAITING	受信要求データ長になるか、タイムアウト時間に達するまで待ちます。

② ***req_length*** は受信要求データ長で、最大１２８バイトです。

③ ****data*** は受信データを格納するバッファのポインタです。

④ ****length*** は受信データ長で、最大１２８バイトです。

⑤ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了です。
１３	TIME_OUT	受信待機中、一定時間受信がありません。

例

データを受信し表示します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    unsigned char buff[ 2048 ];
    short len;
    short i;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST ) )
      {
        if ( NO_ERROR == gets_rs232c( WAITING, 1, buff, &len, 5 ) )
          {
            for ( i = 0; i < len; i++ )
#ifdef __WATCOMC__
                cprintf( "%d \n", buff[ i ] );
#else
                printf( "%d \n", buff[ i ] );
#endif
          }
        finish_rs232c();
      }
  }
```

留意事項

ありません。

３．Ｃ－ＬＩＮＫライブラリ

本ライブラリは、Ｃ－ＬＩＮＫで通信を行うためのものです。

init_clink

機　能

Ｃ－ＬＩＮＫインターフェイスの初期化

形　式

```
#include "clink.h"
short init_clink( short node );
```

解　説

Ｃ－ＬＩＮＫインターフェイスを初期化します。
Ｃ－ＬＩＮＫを使用する前には、必ず、ＲＳ－２３２Ｃの初期化（init_rs232c）後、この関数を呼び出す必要があります。（オープン関数に相当します。）

① ***node*** は自局ノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
０	NO_ERROR	常に正常終了

例

load_clinkを参照してください。

留意事項

Ｃ－ＬＩＮＫを使用する前には、必ず、ＲＳ－２３２Ｃを初期化（init_rs232c）してください。
ＲＳ－２３２Ｃが初期化されていない場合の動作は保証できません。

finish_clink

機　能

Ｃ－ＬＩＮＫインターフェイスの終了処理

形　式

```
#include "clink.h"
short finish_clink( void );
```

解　説

Ｃ－ＬＩＮＫインターフェイスを終了させます。
Ｃ－ＬＩＮＫを終了する前には、必ず、この関数を呼び出す必要があります。
（クローズ関数に相当します。）

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了。
－2	NO_CONNECT	まだ接続されていません。

例

load_clinkを参照してください。

留意事項

この関数を呼び出さないでプログラムの実行を終了した場合、通常のＲＳ－２３２Ｃで制御コード等が使用される場合、ＲＳ－２３２Ｃ通信が正常に動作しない可能性があります。

load_clink

機　能

ファイルの送信

形　式

```
#include "clink.h"
short load_clink(short mode, short node, FILE *fp, char *ver,
                        void (*func)(long totalsize, long size), short rtime);
```

解　説

ファイル（システムファイル、ユーザーファイル）を送信します。

① ***mode*** で送信するファイルの種別を指定します。

値	定　数	意　味
0	SYSTEMFILE	システムファイルの送信を指定します。
1	USERFILE	ユーザファイルの送信を指定します。

② ***node*** は送信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

③ ****fp*** は送信するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"r","b"です。

④ ****ver*** はシステムバージョンの文字列ポインタです。
システムファイルの転送時使用します。
ユーザーファイル時はＮＵＬＬにしてください。

⑤ ****func*** はユーザー定義の関数のポインタです。
送信中に処理したいこと、送信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
　送信データの総バイト数（最初の呼出時のみ有効、以降は０）

size
　今回送信したバイト数

を渡します。
この関数は送信の合間に（約２０４８バイト送信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

⑥ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
6	LINE_BUSY	伝走路がビジィ状態であったため中断しました。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。
１０	FILE_SIZE_ERROR	ファイルのサイズが短すぎます。（１６バイト以下）
１１	FILE_READ_ERROR	ファイル読み込みエラーです。
１３	TIME_OUT	タイムアウトエラーです。

NO_CONNECT,NOT_READY,LINE_BUSYはリトライ後に返します。

例

ファイル"test.bin"を送信します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1 , XCOFF, FIRST )
      {
        if ( NO_ERROR == init_clink( NODE_0 ) )
          {
            if ( ( fp = fopen( "test.bin", "rb" ) ) != NULL )
              {
                load_clink( USERFILE, NODE_8, fp, NULL, NULL, 5 );
              }
            finish_clink();
          }
        finish_rs232c();
      }
  }
```

留意事項

ユーザプログラムでファイルを受信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

upload_clink

機　能

ファイルの受信

形　式

```
#include "clink.h"
short upload_clink(short node, FILE *fp,
                        void (*func)(long totalsize,long size), short rtime);
```

解　説

ファイルを受信します。

① ***node*** は受信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***fp*** は受信したデータを保存するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"w","b"です。

③ ***func*** はユーザー定義の関数のポインタです。
受信中に処理したいこと、受信の進み具合を示すインジケーターなどを行ってください。
この関数への引き数として、

totalsize
　受信データの総数（最初の呼出時のみ有効、以降は０）
size
　今回受信した数

を渡します。
この関数は受信の合間に（約２０４８バイト受信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

④ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。
１２	FILE_WRITE_ERROR	ファイル書き込みエラーです。
１３	TIME_OUT	受信待機中、一定時間受信ありません。

NO_CONNECT,TIME_OUTはリトライ後に返します。

例

バイナリーファイルを受信し、ファイル"test.bin"に保存します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    FILE *fp;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST )
      {
        if ( NO_ERROR == init_clink( NODE_0 ) )
          {
            if ( ( fp = fopen( "test.bin", "wb" ) ) != NULL )
              {
                upload_clink( NODE_8, fp, NULL, 5 );
              }
            finish_clink();
          }
        finish_rs232c();
      }
  }
```

留意事項

ユーザプログラムでファイルを送信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。

outcmd_clink

機　能

システムコマンドの発行

形　式

```
#include "clink.h"
short outcmd_clink(short node, char *s, short rtime);
```

解　説

システムコマンドを発行します。

① ***node*** は発行先のノード番号です

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***s*** はコマンド列のポインタです。ラインターミネータは内部で自動的に付加します。

③ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
6	LINE_BUSY	伝送路がビジィ状態であったため中断しました。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。
１４	CMD_SIZE_ERROR	コマンドの長さが2048を越えています。

NO_CONNECT,NOT_READY,LINE_BUSYはリトライ後に返します。

例

ファイル操作モードに移行します。

```
#include <stdio.h>
#include <string.h>
#include "clink.h"

#define RETRY_TIME    5

void main( void )
  {
    char cmdbuff[ 10 ];

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST )
      {
        if ( NO_ERROR == init_clink( NODE_0 ) )
          {
            strcpy( cmdbuff, "F" );  /* ファイル操作モード移行コマンド */
            if ( NO_ERROR == outcmd_clink( NODE_8, cmdbuff, RETRY_TIME ) )
              {
                if ( NO_ERROR == checkcmdack_clink( NODE_8, RETRY_TIME ) )
                  {
                    if ( NO_ERROR == checkprmpt_clink( NODE_8, RETRY_TIME ) )
#ifdef __WATCOMC__
                      cprintf( "ファイル操作モードに移行成功\n" );
#else
                      printf( "ファイル操作モードに移行成功\n" );
#endif
                  }
              }
            finish_clink();
          }
        finish_rs232c();
      }
  }
```

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkcmdack_clink

機　能

コマンド応答チェック

形　式

```
#include "clink.h"
short checkcmdack_clink(short node, short rtime);
```

解　説

システムコマンドの応答をチェックしCMD_ACK('!'), CMD_NAK('%'), CMD_NON('?')が返されるまで指定時間待ちます。

CMD__ＡＣＫ ‥‥‥‥‥ コマンド処理結果が正常です。
CMD__ＮＡＫ ‥‥‥‥‥ コマンド処理結果が異常です。
CMD__ＮＯＮ ‥‥‥‥‥ 未定義コマンドです。

① ***node*** は送受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。 （CMD_ACKが返されました）
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
３７	CMD_NAK	CMD_NAKが返されました。
６３	CMD_NON	CMD_NONが返されました。

TIME_OUT,NO_CONNECTはリトライ後返します。

例

outcmd_clinkを参照してください。

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

checkprmpt_clink

機　能

プロンプトチェック

形　式

```
#include "clink.h"
short checkprmpt_clink(short node, short rtime);
```

解　説

システムプロンプト('>')が返されるまで指定時間待ちます。

① ***node*** 送受信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
０	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。

TIME_OUT,NO_CONNECTはのリトライ後返します。

例

outcmd_clinkを参照してください。

留意事項

システムプロンプトについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

puts_clink

機　能

データの送信

形　式

```
#include "clink.h"
short puts_clink( unsigned char *data, short length,
                                        short type,short node, short rtime );
```

解　説

指定されたノード番号にデータを送信します。

① ***data*** は送信データが格納されているバッファのポインタです。

② ***length*** は送信データ長（１から２０４８）です。

③ ***type*** は送信データの型です。

値	定　　数	意　　味
0	TEXT_TYPE	アスキーです。
1	BINARY_TYPE	バイナリです。

④ ***node*** は送信先のノード番号です。

値	定　　数	意　　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　　数	意　　味
0	NO_ERROR	正常終了しました。
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
1	NOT_READY	送信先のノードが受信セットアップされていませんでした。
6	LINE_BUSY	伝走路がビジィ状態であったため中断しました。
１３	TIME_OUT	タイムアウトエラーです。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。

NO_CONNECT,NOT_READY,LINE_BUSYはリトライ後返します。

例

データ0x00,0x01,0x02,0x03,0x04を送信します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    unsigned char buff[ 2048 ];
    short i;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST )
      {
        if ( NO_ERROR == init_clink( NODE_0 ) )
          {
            for ( i = 0; i < 5; i++ )
                buff[ i ] = i;
            puts_clink( buff, i, BINARY_TYPE, NODE_8, 5 );
            finish_clink();
          }
        finish_rs232c();
      }
  }
```

留意事項

９０Ｘ 基本ライブラリのLib_puts_clinkと、引き数、戻り値、型が違います。
注意してください。

gets_clink

機　能

データの受信

形　式

```
#include "clink.h"
short gets_clink( unsigned char *data, short *length,
                                        short *type, short *node, short rtime );
```

解　説

指定されたノード番号からデータを受信します。

① ***data*** は受信データを格納するバッファのポインタです。

② ***lengt h*** は受信データ長（１から２０４８）です。

③ ***type*** は受信データの型です。

値	定　数	意　味
0	TEXT_TYPE	アスキーです。
1	BINARY_TYPE	バイナリです。

④ ***node*** は送信元のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。

戻り値

値	定　数	意　味
0	NO_ERROR	正常終了しました。
－２	NO_CONNECT	まだ接続されていません。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
８８８	HW_ERROR	インタフェイスまたは制御関数に異常が発生しました。

NO_CONNECT,TIME_OUTはリトライ後返します。

例

データを受信し、表示します。

```
#include <stdio.h>
#include "clink.h"

void main( void )
  {
    unsigned char buff[ 2048 ];
    short len, type, node;
    short i;

    if ( NO_ERROR == init_rs232c( PC9801, B9600, L8, PNONE, S1, XCOFF, FIRST )
      {
        if ( NO_ERROR == init_clink( NODE_0 ) )
          {
            gets_clink( buff, &len, &type, &node, 5 );
            printf( "長さ：%d  型：%d  ノード：%d \n", len, type, node );
            for ( i = 0; i < len; i++ )
#ifdef __WATCOMC__
                cprintf(  "%d \n", buff[ i ] );
#else
                printf( "%d \n", buff[ i ] );
#endif
            finish_clink();
          }
        finish_rs232c();
      }
  }
```

留意事項

９０Ｘ 基本ライブラリのLib_gets_clinkと、引き数、戻り値、型が違います。
注意してください。

４．サンプルプログラム

4.1　MS-DOS16ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ(Turbo C++/Microsoft C編)

ＭＳ－ＤＯＳ１６ビット版ホストＣ－ＬＩＮＫライブラリを使用したサンプルプログラムが

\sample\ex_clink.c ･･･････ サンプルプログラム
\sample\ex_clink.exe ･････ turbocl.batを使用し作成した実行ファイル
turbocl.bat ･･･････････････ ex_clink.cのコンパイル、リンクを行うバッチファイル
(Turbo C++用)
mscl.bat ･･･････････････････ ex_clink.cのコンパイル、リンクを行うバッチファイル
(Microsoft C用)

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

ex_clink [option [option...]] filename

① ***-Mn*** ･･･････････････ ホストのノード番号です。ｎは０～７で、省略時は０です。

② ***-Tn*** ･･･････････････ ターゲットのノード番号です。ｎは８～６３で、省略時は８です。

③ ***filename*** ･･･････････ ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

注意

このプログラムはＰＣ９８０１用で、ホスト、ＣＳＣ９０Ｘ供、ＲＳ２３２ＣのＣＨ０を使用しています。

4.2　MS-DOS16ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ(Wactom C/C++編)

ＭＳ－ＤＯＳ１６ビット版ホストＣ－ＬＩＮＫライブラリを使用したサンプルプログラムが

\sample\lw_clink.c ･･･････サンプルプログラム
\sample\lw_clink.exe ･････wtcl.batを使用し作成した実行ファイル
wtcl.bat ･･･････････････････lw_clink.cのコンパイル、リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

lw_clink [option [option...]] filename

① ***-Hn*** ･･･････････ホストの機種番号です。
ｎは0(PC9801), 1(IBMPC), 2(J3100), 3(AX) 省略時は０です。

② ***-Bn***･･･････････ボーレートです。
ｎは1200, 1800, 2400, 4800, 9600, 19200 省略時は9600です。

③ ***-Ln***･･･････････データ長です。ｎは5, 6, 7, 8 省略時は８です。

④ ***-Pn***･･･････････パリティです。ｎは0(NONE), 1(ODD), 3(EVEN) 省略時は０です。

⑤ ***-Sn***･･･････････ストップビットです。
ｎは0(1bits), 1(1.5bits), 3(2bits) 省略時は０です。

⑥ ***-Mn*** ･･･････････ホストのノード番号です。
ｎは０～７で、省略時は０です。

⑦ ***-Tn***･･･････････ターゲットのノード番号です。
ｎは８～６３で、省略時は８です。

⑧ ***filename*** ･･･････ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

注意

このプログラムはホスト、ＣＳＣ９０Ｘ供、ＲＳ２３２ＣのＣＨ０を使用しています。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcl.bat を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-16-BIT
[Image Type] を Executable [.exe]
にしてください。
スタックサイズの指定、メモリモデルの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

4.3　ＭＳ－ＤＯＳ３２ビット用サンプルプログラム

ＭＳ－ＤＯＳ３２ビット版ホストＣ－ＬＩＮＫライブラリを使用したサンプルプログラムが

\sample\xw_clink.c ･･･････サンプルプログラム
\sample\xw_clink.exe ･････wtcl32.batを使用し作成した実行ファイル
wtcl32.bat ･･･････････････xw_clink.cのコンパイル、リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０ＸのＦＳ０にファイルをコピーするものです。

使い方

xw_clink [option [option...]] filename

① ***-Hn*** ･･･････････････ホストの機種番号です。
ｎは0(PC9801), 1(IBMPC), 2(J3100), 3(AX) 省略時は０です。

② ***-Bn***･･･････････････ボーレートです。
ｎは1200, 1800, 2400, 4800, 9600, 19200 省略時は9600です。

③ ***-Ln***･･･････････････データ長です。ｎは5, 6, 7, 8 省略時は８です。

④ ***-Pn***･･･････････････パリティです。ｎは0(NONE), 1(ODD), 3(EVEN) 省略時は０です。

⑤ ***-Sn***･･･････････････ストップビットです。
ｎは0(1bits), 1(1.5bits), 3(2bits) 省略時は０です。

⑥ ***-Mn*** ･･･････････････ホストのノード番号です。ｎは０～７で、省略時は０です。

⑦ ***-Tn***･･･････････････ターゲットのノード番号です。n は８～６３で、省略時は８ です。

⑧ ***filename*** ･･･････ＣＳＣ９０Ｘ用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

注意

このプログラムはホスト、ＣＳＣ９０Ｘ供、ＲＳ２３２ＣのＣＨ０を使用しています。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcl32.bat を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-32-BIT
[Image Type] を DOS4GW Executable [.exe]
にしてください。
スタックサイズの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

第３編

ＤＯＳ版
ホストパラレル・リンクライブラリ

アルファベット順ライブラリ目次

項番	ライブラリ名	ページ
1	checkcmdack_par_link	130
2	checkprmpt_par_link	131
3	finish_par_link	123
4	gets_par_link	136
5	init_par_link	122
6	last_error_par_link	140
7	load_par_link	124
8	outcmd_checkcmdack_par_link	132
9	outcmd_par_link	128
10	par_last_error	138
11	puts_par_link	134
12	upload_par_link	126

１．概　　要

本ライブラリは、ＣＳＣ９０２とホスト間をパラレル・リンクによる通信を行うライブラリです。
（ＣＳＣ９０１シリーズではサポートされていません）
本ライブラリには、ＭＳ－ＤＯＳ１６ビット版とＭＳ－ＤＯＳ３２ビット版が用意されています。

１６ビット版と３２ビット版については、各ライブラリの仕様、インクルードファイルは、共通となっています。

ＭＳ－ＤＯＳ１６ビット版ホストパラレル・リンクライブラリ

動作環境

ホスト	：ＮＥＣ ＰＣ－９８０１シリーズ ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ	：ＭＳ－ＤＯＳ
コンパイラ	：Ｔｕｒｂｏ Ｃ++ Ｖｅｒ．１．０ Ｍｉｃｒｏｓｏｆｔ Ｃ Ｖｅｒ．６．０ Ｗａｔｃｏｍ Ｃ／Ｃ++ Ｖｅｒ．１１．０Ｊ

構成

\lib\ltplink.lib‥‥‥‥‥‥Turbo C++ Ver. 1.0用ライブラリ
\lib\lmplink.lib‥‥‥‥‥‥Microsoft C Ver. 6.0用ライブラリ
\lib\lwplink.lib‥‥‥‥‥‥Watcom C/C++ Ver.11.0J用ライブラリ
\include\par_link.h‥‥‥‥パラレル・リンクライブラリ用インクルードファイル

注意

◎ＰＣ－９８０１シリーズでは、双方向フルセントモードをサポートしている機種（３６pin ハーフピッチコネクタがついているもの：ＰＣ－９８２１Ａｐ２以降）のみサポートしています。
◎本ライブラリはラージモデル用です。
◎本ライブラリはプログラムのスタックを多く使用しています。
スタックサイズを 4096 以上にしてください。
Microsoft C の場合はリンカオプション/STACK でスタックサイズを 4096 以上にします。
Turbo C++の場合はデフォルトのスタックサイズは 4096 です。
4096 以上の指定する場合はグローバル変数_stklen で指定します。
Watcom C/C++の場合はリンカオプション STACK でスタックサイズを 4096 以上にします。
コンパイル、リンクの参考として**３．サンプルプログラム**の項目を参照してください。
◎Windows の MS-DOS プロンプトでの動作はサポートされません。

ＭＳ－ＤＯＳ３２ビット版ホストパラレル・リンクライブラリ

動作環境

ホスト	：ＮＥＣ ＰＣ－９８０１シリーズ 　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ	：ＭＳ－ＤＯＳ
コンパイラ	：Ｗａｔｃｏｍ Ｃ／Ｃ＋＋ Ｖｅｒ．１１．０Ｊ

構成

\lib\xwplink.lib ・・・・・・・・・Watcom C/C++ Ver.11.0J用ライブラリ
\include\par_link.h ・・・・・・パラレル・リンクライブラリ用インクルードファイル

注意

◎本ライブラリはプログラムのスタックを多く使用しています。
　スタックサイズを 4096 以上にしてください。
　リンカオプション STACK でスタックサイズを 4096 以上にします。
　コンパイル、リンクの参考として**３．サンプルプログラム**の項目を参照してください。
◎Windows の MS-DOS プロンプトでの動作はサポートされません。

２．パラレル・リンクライブラリ

本ライブラリは、パラレル・リンクで通信を行うためのものです。

init_par_link

機　能

パラレル・リンクインターフェイスの初期化

形　式

```
#include "Par_Link.h"
short init_par_link( short machine, short ch );
```

解　説

パラレル・リンクインターフェイスを初期化します。（オープン関数に相当します。）
パラレル・リンクを使用する前には、必ずコールする必要があります。

① ***machine*** は使用するホストマシンです。

値	定　数	意　味
0	P_PCAT	ＩＢＭＰＣおよび互換機。
1	P_PC98	ＰＣ９８シリーズおよび互換機。

② ***ch*** はパラレルポートのチャネル番号です。（ＩＢＭＰＣおよび互換機のみ）

値	定　数	意　味
0	P_CH_0	ポートアドレス378h～37Ahを指定します。
1	P_CH_1	ポートアドレス278h～27Ahを指定します。
2	P_CH_2	ポートアドレス3BCh～3BEhを指定します。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

load_par_linkを参照してください。

留意事項

ありません。

finish_par_link

機　能

パラレル・リンクインターフェイスの終了処理

形　式

```
#include "Par_Link.h"
short finish_par_link( void );
```

解　説

パラレル・リンクインターフェイスを終了させます。（クローズ関数に相当します。）
パラレル・リンクを終了する前には、必ずコールする必要があります。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

load_par_linkを参照してください。

留意事項

ありません。

load_par_link

機　能

ファイルの送信

形　式

```
#include "Par_Link.h"
short load_par_link( FILE fp, void(*func)
                                    (long totalsize, lonh size), short rtime );
```

解　説

ファイル（システムファイル、ユーザーファイル）を送信します。

① ***fp*** は送信するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"r", "b"です。

② ***func*** はユーザー定義の関数のポインタです。
送信中に処理したいこと、送信の進み具合を示すインジケータなどを行ってください。
この関数の引き数として、

totalsize
送信データの総バイト数（最初の呼出時のみ有効、以降０）

size
今回送信したバイト数

を渡します。
この関数は送信の合間（約２０４８バイト送信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

③ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

ファイル"test.exe"を送信します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        FILE    *fp;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                if( ( fp = fopen( "test.exe", "rb" ) ) != NULL )
                {
                        load_par_link( fp, NULL, 5 );
                        fclose( fp );
                }
                finish_par_link();
        }
}
```

留意事項

ユーザプログラムでファイルで受信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
０	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－２０	PE_FILE_SIZE	ファイルのサイズが短いです。
－２１	PE_FILE_R	ファイル読み込みエラーです。
－２２	PE_FILE_W	ファイル書き込みエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

upload_par_link

機　能

ファイルの受信

形　式

```
#include "Par_Link.h"
short upload_par_link( FILE fp, void(*func)(long totalsize, lonh size),
                                                    short rtime );
```

解　説

ファイルを受信します。

① ***fp*** は受信したデータを保存するファイルのファイルポインタです。
fopen関数が返すストリームのポインタで、アクセス形式は"w", "b"です。

② ***func*** はユーザー定義の関数のポインタです。
受信中に処理したいこと、受信の進み具合を示すインジケータなどを行ってください。
この関数の引き数として、

totalsize
受信データの総バイト数（最初の呼出時のみ有効、以降０）

size
今回受信したバイト数

を渡します。
この関数は受信の合間（約２０４８バイト受信毎）呼び出されます。
（関数のポインタとしてＮＵＬＬが指定されると呼び出しません。）

③ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

ファイルを受信しファイル”test.exe”に保存します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        FILE    *fp;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                if( ( fp = fopen( "test.exe", "wb" ) ) != NULL )
                {
                        upload_par_link( fp, NULL, 5 );
                        fclose( fp );
                }
                finish_par_link();
        }
}
```

留意事項

ユーザプログラムでファイルで送信するには、９０Ｘ 基本ライブラリのLib_fcopyを使用します。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
０	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－２０	PE_FILE_SIZE	ファイルのサイズが短いです。
－２１	PE_FILE_R	ファイル読み込みエラーです。
－２２	PE_FILE_W	ファイル書き込みエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

outcmd_par_link

機　能

システムコマンドの発行

形　式

```
#include "Par_Link.h"
short outcmd_par_link( char *cmd_buf, short rtime );
```

解　説

システムコマンドを発行します。

① ***cmd_buf*** はコマンド列のポインタで、ラインターミネータは内部で自動的に付加します。

② ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

ファイル操作モードに移行します。

```
#include <stdio.h>
#include <string.h>
#include "par_link.h"

#define RETRY_TIME      10

void    main()
{
        char    cmdbuff[10];

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                strcpy( cmdbuff, "F" );     /* ファイル操作モード移行コマンド */
                if( outcmd_par_link( cmdbuff, RETRY_TIME ) )
                {
                        if( checkcmdack_par_link( RETRY_TIME ) )
                        {
                                if( checkprmpt_par_link( RETRY_TIME ) )
#ifdef __WATCOMC__
                                  cprintf( "ファイル操作モードに移行成功\n" );
#else
                                  printf( "ファイル操作モードに移行成功\n" );
#endif
                                }
                        }
                        finish_par_link();
                }
}
```

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
０	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

checkcmdack_par_link

機　能

コマンド応答チェック

形　式

```
#include "Par_Link.h"
short checkcmdack_par_link( short rtime );
```

解　説

システムコマンドの応答をチェックし、ＡＣＫ（‘ ！ ’）が返されるまで指定時間待ちます。

① ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

outcmd_par_linkを参照してください。

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　数	意　味
0	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－３１	PE_CMD_NAK	ＮＡＫ（‘ ％ ’）が返されました。
－３２	PE_NOT_OPEN	ＮＯＮ（‘ ？ ’）が返されました。
－９９	PE_NOT_OPEN	まだ接続されていません。

checkprmpt_par_link

機　能

プロンプトチェック

形　式

```
#include "Par_Link.h"
short checkprmpt_par_link( short rtime );
```

解　説

システムプロンプト（‘＞’）が返されるまで指定時間待ちます。

① ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

outcmd_par_linkを参照してください。

留意事項

システムプロンプトについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　数	意　味
0	なし	エラーは発生していない。
－2	PE_TIME_OUT	タイムアウトエラーです。
－3	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－4	PE_PARM	引数の値が異常です。
－5	PE_CSC	ＣＳＣチェックエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

outcmd_checkcmdack_par_link

機　能

システムコマンドの発行＆コマンド応答のチェック

形　式

```
#include "Par_Link.h"
short outcmd_checkcmdack_par_link( char *cmd_buf, short rtime );
```

解　説

システムコマンドを発行して応答のチェックを行います。
この時、コマンドが返されるまで指定時間待ちます。

① ***cmd_buf*** はコマンド列のポインタです。ラインターミネータは内部で自動的に付加します。

② ***rtime*** はリトライタイム（秒単位）です。
リトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	P_ACK	ＡＣＫ（‘！’）が返されました。
2	P_NAK	ＮＡＫ（‘％’）が返されました。
3	P_NON	ＮＯＮ（‘？’）が返されました。
0	なし	異常終了しました。

例

ファイル操作モードに移行します。

```
#include <stdio.h>
#include <string.h>
#include "par_link.h"

#define RETRY_TIME      10

void    main()
{
        char    cmdbuff[10];

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                strcpy( cmdbuff, "F" );         /* ファイル操作モード移行コマンド */
                if( P_ACK == outcmd_checkcmdack_par_link( cmdbuff, RETRY_TIME ) )
                {
                        if( checkprmpt_par_link( RETRY_TIME ) )
#ifdef __WATCOMC__
                          cprintf( "ファイル操作モードに移行成功\n" );
#else
                          printf( "ファイル操作モードに移行成功\n" );
#endif
                }
                finish_par_link();
        }
}
```

留意事項

システムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。
異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
０	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
９９	PE_NOT_OPEN	まだ接続されていません。

puts_par_link

機　能

データの送信

形　式

```
#include "Par_Link.h"
short puts_par_link( char *buf_adr, short length, short rtime );
```

解　説

データを送信します。

① ***buf_adr*** は送信データが格納されているバッファのポインタです。

② ***length*** は送信データ長で、最大２０４８バイトです。

③ ***rtime*** はリトライタイム（秒単位）です。
送信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

データ0x00,0x01,0x02,0x03,0x04を送信します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        char    buff[10];
        int             i;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                for( i = 0; i < 5; i++ )
                        buff[i] = i;
                puts_par_link( buff, i, 10 );
                finish_par_link();
        }
}
```

留意事項　異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
0	なし	エラーは発生していない。
－2	PE_TIME_OUT	タイムアウトエラーです。
－3	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－4	PE_PARM	引数の値が異常です。
－5	PE_CSC	ＣＳＣチェックエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

gets_par_link

機　能

データの受信

形　式

```
#include "Par_Link.h"
short gets_par_link( char *buf_adr, short length, short rtime );
```

解　説

データを受信します。

① ***buf_adr*** は受信データを格納するバッファのポインタです。

② ***length*** は受信データ長で、最大２０４８バイトです。

③ ***rtime*** はリトライタイム（秒単位）です。
受信中にリトライ可能なエラーが発生した場合、指定された時間内でリトライします。
０～６０までの値を指定してください。

戻り値

処理結果

値	定　数	意　味
1	なし	正常終了しました。
0	なし	異常終了しました。

例

データを受信し表示します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        char    buff[2048];
        short   len;
        int     i;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                if( gets_par_link( buff, &len, 10 ) )
                {
                        for( i = 0; i < len; i++ )
#ifdef __WATCOMC__
                            cprintf(  "%d\n", buff[i] );
#else
                            printf( "%d\n", buff[i] );
#endif
                        }
                        finish_par_link();
                }
}
```

留意事項　異常終了した場合、par_last_error関数でエラー情報を取得できます。

値	定　　数	意　　味
0	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－９９	PE_NOT_OPEN	まだ接続されていません。

par_last_error

機　能

エラー情報の取得１

形　式

```
#include "Par_Link.h"
short par_last_error( void );
```

解　説

最後に発生したエラー情報を取得します。
パラレル･リンク独自のエラー値を返します。

戻り値

処理結果

値	定　　数	意　　味
０	なし	エラーは発生していない。
－２	PE_TIME_OUT	タイムアウトエラーです。
－３	PE_NOT_CONNECT	まだ接続されていません。 （その他ハード的なエラー）
－４	PE_PARM	引数の値が異常です。
－５	PE_CSC	ＣＳＣチェックエラーです。
－２０	PE_FILE_SIZE	ファイルのサイズが短いです。
－２１	PE_FILE_R	ファイル読み込みエラーです。
－２２	PE_FILE_W	ファイル書き込みエラーです。
－３１	PE_CMD_NAK	ＮＡＫ（‘％’）が返されました。
－３２	PE_CMD_NON	ＮＯＮ（‘？’）が返されました。
－９９	PE_NOT_OPEN	まだ接続されていません。

例

エラーが発生した場合、エラー内容を取得します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        FILE    *fp;
        short   ret;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                if( ( fp = fopen( "SAMPLE.D91", "rb" ) ) != NULL )
                {
                        if( !load_par_link( fp, NULL, 5 ) )
                        {
                                ret = par_last_error();
#ifdef __WATCOMC__
                                cprintf( "ERROR:%d\n" ret );
#else
                                printf( "ERROR:%d\n" ret );
#endif
                        }
                        fclose( fp );
                }
                finish_par_link();
        }
}
```

留意事項

ありません。

last_error_par_link

機　能

エラー情報の取得２

形　式

```
#include "Par_Link.h"
short last_error_par_link( void );
```

解　説

最後に発生したエラー情報を取得します。
ホストＣ－ＬＩＮＫライブラリ、ホストＣ－ＮＥＴライブラリと同様のエラー値を返します。

戻り値

処理結果

値	定　　数	意　　味
０	NO_ERROR	エラーは発生していない。
－１	ARG_ERR	PE_PARM(-4)。
－２	NO_CONNECT	PE_NOT_OPEN(-99)。
１０	FILE_SIZE_ERROR	PE_FILE_SIZE(-20)。
１１	FILE_READ_ERROR	PE_FILE_R(-21)。
１２	FILE_WRITE_ERROR	PE_FILE_W(-22)。
１３	TIME_OUT	PE_TIME_OUT(-2)。
８８８	HW_ERROR	PE_NOT_CONNECT(-3),PE_CSC(-5)。

例

エラーが発生した場合、エラー内容を取得します。

```
#include <stdio.h>
#include "par_link.h"

void    main()
{
        FILE    *fp;
        short   ret;

        if( init_par_link( P_PCAT, P_CH_0 ) )
        {
                if( ( fp = fopen( "SAMPLE.DAT", "wb" ) ) != NULL )
                {
                        if( !upload_par_link( fp, NULL, 5 ) )
                        {
                                ret = last_error_par_link();
#ifdef __WATCOMC__
                                cprintf( "ERROR:%d\n", ret );
#else
                                printf( "ERROR:%d\n", ret );
#endif
                        }
                        fclose( fp );
                }
                finish_par_link();
        }
}
```

留意事項

ありません。

３．サンプルプログラム

3.1　MS-DOS16ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ(Turbo C++/Microsoft C 編)

ＭＳ－ＤＯＳ１６ビット版ホストパラレル・リンクライブラリを使用したサンプルプログラムが

\sample\ex_plink.c ‥‥‥ サンプルプログラム
\sample\ex_plink.exe ‥‥‥ turbocp.batを使用し作成した実行ファイル
turbocp.bat ‥‥‥‥‥‥‥‥ ex_plink.cのコンパイル、リンクを行うバッチファイル
(Turbo C++用)
mscp.bat ‥‥‥‥‥‥‥‥‥‥ ex_plink.cのコンパイル、リンクを行うバッチファイル
(Microsoft C用)

として、あります。
このプログラムは、ホストからＣＳＣ９０２のＦＳ０にファイルをコピーするものです。

使い方

ex_clink [option [option...]] filename

① ***-Mn*** ‥‥‥‥‥‥‥‥‥‥‥ ホストの機種番号です。
ｎは０：PC/AT, １：PC98で、省略時は０です。

② ***-Pn*** ‥‥‥‥‥‥‥‥‥‥‥ パラレルポートのチャネル番号です。
ｎは０：３７８ｈ～３７Ａｈ, １：２７８ｈ～２７Ａｈ,
２：３ＢＣｈ～３ＢＥｈで、省略時は０です。

③ ***filename*** ‥‥‥‥‥‥‥‥ ＣＳＣ９０２用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

3.2　MS-DOS16ﾋﾞｯﾄ用ｻﾝﾌﾟﾙﾌﾟﾛｸﾞﾗﾑ(Wactom C/C++編)

ＭＳ－ＤＯＳ１６ビット版ホストパラレル・リンクライブラリを使用したサンプルプログラムが

\sample\lw_plink.c ･･･････ サンプルプログラム
\sample\lw_plink.exe ･････ wtcp.batを使用し作成した実行ファイル
wtcp.bat ･･･････････････････ lw_clink.cのコンパイル、リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０２のＦＳ０にファイルをコピーするものです。

使い方

lw_clink [option [option...]] filename

① ***-Mn*** ･･･････････････ ホストの機種番号です。
ｎは０：PC/AT、１：PC98で、省略時は０です。

② ***-Pn*** ･･･････････････ パラレルポートのチャネル番号です。
ｎは０：３７８ｈ～３７Ａｈ，１：２７８ｈ～２７Ａｈ，
２：３ＢＣｈ～３ＢＥｈで、省略時は０ です。

③ ***filename*** ･･･････ ＣＳＣ９０２用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcp.bat を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-16-BIT
[Image Type] を Executable [.exe]
にしてください。
スタックサイズの指定、メモリモデルの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

3.3　ＭＳ－ＤＯＳ３２ビット用サンプルプログラム

ＭＳ－ＤＯＳ３２ビット版ホストパラレル・リンクライブラリを使用したサンプルプログラムが

\sample\xw_plink.c ･･･････ サンプルプログラム
\sample\xw_plink.exe ･････ wtcp32.batを使用し作成した実行ファイル
wtcp32.bat ･･･････････････ lw_clink.cのコンパイル、リンクを行うバッチファイル

として、あります。
このプログラムは、ホストからＣＳＣ９０２のＦＳ０にファイルをコピーするものです。

使い方

xw_clink [option [option...]] filename

① ***-Mn*** ･･･････････････ ホストの機種番号です。
ｎは０：PC/AT, １：PC98で、省略時は０です。

② ***-Pn*** ･･･････････････ -Pnパラレルポートのチャネル番号です。
ｎは０：３７８ｈ～３７Ａｈ, １：２７８ｈ～２７Ａｈ,
２：３ＢＣｈ～３ＢＥｈで、省略時は０ です。

③ ***filename*** ･･･････ ＣＳＣ９０２用のコピー元のファイル名です。

引き数を全て省略すると使い方が表示されます。

コンパイル、リンクの一連手順

①バッチファイルの使用
wtcp32.bat を使用します。
コンパイルオプション、リンクオプションについての詳細は「Watcom User's Guide」を参照してください。

②ＷＡＴＣＯＭ統合開発環境の使用
Ｗｉｎｄｏｗｓ環境での開発は、ＷＡＴＣＯＭ統合開発環境（ＩＤＥ）を使用してプログラムの作成ができます。
新規にプロジェクトを作成する時に
[Traget Environment] を DOS-32-BIT
[Image Type] を DOS4GW Executable [.exe]
にしてください。
スタックサイズの指定以外コンパイラとリンカのオプションは基本的に変更せずにご使用なれます。

第４編

Ｗｉｎｄｏｗｓ９５／ＮＴ版
ホスト通信ライブラリ

アルファベット順ライブラリ目次

項番	ライブラリ名	ページ
1	CSCCommCheckCmdAck	189
2	CSCCommCheckSeq	192
3	CSCCommFinish	176
4	CSCCommGetClinkParam	168
5	CSCCommGetCnetParam	163
6	CSCCommGetErrorNo	205
7	CSCCommGetNodeMap	177
8	CSCCommGetParlinkParam	172
9	CSCCommGetRs232cParam	158
10	CSCCommGets	179
11	CSCCommGets2	181
12	CSCCommInit	174
13	CSCCommLoad	199
14	CSCCommOutCmd	186
15	CSCCommOutComand	190
16	CSCCommPuts	183
17	CSCCommSendWatch	196
18	CSCCommSetClinkParam	165
19	CSCCommSetCnetParam	160
20	CSCCommSetParlinkParam	170
21	CSCCommSetRs232cParam	156
22	CSCCommUpLoad	202
23	CSCCommWatch	193

１．概　　要

本ライブラリは、ＣＳＣ９０Ｘとホスト間の通信を行うライブラリで、以下の通信デバイスを用意しています。

☆ＲＳ－２３２Ｃ
☆Ｃ－ＮＥＴ
☆Ｃ－ＬＩＮＫ
☆パラレル・リンク
（MS-WindowsNTでは、C-LINK、パラレル・リンクはサポートされていません）

動作環境

ホスト　　：ＮＥＣ ＰＣ－９８０１シリーズ
　　ＩＢＭ ＰＣ／ＡＴ，ＰＣ／ＡＴ互換機
ＯＳ　　：ＭＳ－Ｗｉｎｄｏｗｓ９５／ＭＳ－ＷｉｎｄｏｗｓＮＴ４．Ｘ
コンパイラ　：Ｍｉｃｒｏｓｏｆｔ　Ｖｉｓｕａｌ　Ｃ++ Ｖｅｒ．２．０

WindowsNTの場合は、[スタート]－[設定]－[コントロールパネル]でシリアル・ポートを選択し、COM1/COM2の設定があるかどうかを確認する必要があります。さらに、ＡＴ互換機の場合は、[COM1/2設定]－[詳細]で、以下の設定になっているかどうかも確認する必要があります。

	ＣＯＭ１	ＣＯＭ２
ＣＯＭポート番号	１	２
Ｉ／Ｏポートアドレス	３ｆ８	２ｆ８
割り込み番号（ＩＲＱ）	４	３

構成

本ライブラリを使用するにあたって、以下のファイルが必要です。

＜ＰＣ－９８用＞

CSC_C98 .DLL ‥‥‥‥‥‥‥‥‥‥ 通信用ＤＬＬファイル
\lib\CSC_C98 .LIB ‥‥‥‥‥‥‥‥ 通信用ＬＩＢファイル
\include\CSC_CTRL.H ‥‥‥‥‥‥‥ 通信用インクルードファイル

＜ＰＣ／ＡＴ用＞

CSC_COMM.DLL ‥‥‥‥‥‥‥‥‥‥ 通信用ＤＬＬファイル
\lib\CSC_COMM.LIB ‥‥‥‥‥‥‥‥ 通信用ＬＩＢファイル
\include\CSC_CTRL.H ‥‥‥‥‥‥‥ 通信用インクルードファイル

MS-WindowsNTの場合は、以下のファイルも必要です。

WinRT.sys ‥‥‥‥‥‥‥‥‥‥‥‥‥ WinRTドライバー（SYSTEM32\DRIVERSの下）

通信用ＬＩＢファイルは、リンク時に組み込む必要があります。
通信用ＤＬＬファイルは、一般的に作成されたアプリケーションソフトと同じフォルダ内に格納します。

注意

MS-WindowsNTで使用する場合は、以降の説明では、Windows95をWindowsNTに読み替えて参照してください。
「WindowsNT対応ホスト通信ライブラリ」をインストール際に、事前に「WindowsNT対応ホスト通信プログラム」をインストールしておく必要があります。
ＷｉｎｄｏｗｓＮＴでは、「WinRT」を使用しています。
「WinRT」とは、ドライバー作成ツールで「WinRT」を組み込むことにより通信機能が実現されます。

２．通信ライブラリ

本ライブラリは、ＲＳ－２３２Ｃ，Ｃ－ＮＥＴ，Ｃ－ＬＩＮＫ（Windows95のみ），パラレル・リンク(Windows95のみ)で通信を行うためのものです。

必要なインクルードファイルは、CSC_CTRL.H です。

CSCCommSetRs232cParam

機　能

ＲＳ－２３２Ｃポートの設定

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommSetRs232cParam( BYTE bPort, BYTE bByteSize,  BYTE bFlowCtrl,
                            BYTE bParity, BYTE bStopBits, DWORD dwBaudRate );
```

解　説

ＲＳ－２３２Ｃの各パラメータを設定します。

① ***bPort*** はＲＳ－２３２Ｃのポートです。

値	定　数	意　味
1	FIRST	チャンネル０を使用します。
2	SECOND	チャンネル１を使用します。

② ***bByteSize*** はデータ長です。

値	定　数	意　味
5	L5	５ｂｉｔｓです。
6	L6	６ｂｉｔｓです。
7	L7	７ｂｉｔｓです。
8	L8	８ｂｉｔｓです。

③ ***bFlowCtrl*** はＸフローコントロールです。

値	定　数	意　味
0	XCOFF	フロー制御を行いません。
4	XCON	フロー制御を行います。

④ ***bParity*** はパリティです。

値	定　数	意　味
0	PNONE	パリティなしです。
1	PODD	奇数パリティです。
2	PEVEN	偶数パリティです。

⑤ ***bStopBits*** はストップビットです。

値	定　数	意　味
0	Ｓ１	１ｂｉｔｓです。
1	Ｓ１５	１．５ｂｉｔｓです。
2	Ｓ２	２ｂｉｔｓです。

⑥ ***dwBaudRate*** は通信速度です。

値	定　数	意　味
300	B300	300bpsです。
600	B600	600bpsです。
1200	B1200	1200bpsです。
2400	B2400	2400bpsです。
4800	B4800	4800bpsです。
9600	B9600	9600bpsです。
19200	B19200	19200bpsです。
38400	B38400	38400bpsです。
57600	B57600	57600bpsです。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

ＲＳ－２３２Ｃによる通信を開始します。

```
void func()
{
        INT iComhandle;

        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
        {
                iComHandle = CSCCommInit();   /* ＲＳ－２３２Ｃによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

CSCCommInit関数とペアで使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　数	意　味
－１	ARG_ERR	引数の値が異常です。
２０	CONNECT_ERROR	接続中（ｲﾝﾀｰﾌｪｲｽは初期化済み）です。

CSCCommGetRs232cParam

機　能

ＲＳ－２３２Ｃポートの参照

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommGetRs232cParam( BYTE bPort, BYTE *bByteSize,  BYTE *bFlowCtrl,
                            BYTE *bParity, BYTE *bStopBits,
                            DWORD *dwBaudRate );
```

解　説

ＲＳ－２３２Ｃの各パラメータを参照します。

① ***bPort*** はＲＳ－２３２Ｃのポートで、FIRST(0)/SECOND(1)とする。

② ****bByteSize*** はデータ長です。

③ ****bFlowCtrl*** はＸフローコントロールです。

④ ****bParity*** はパリティです。

⑤ ****bStopBits*** はストップビットです。

⑥ ****dwBaudRate*** は通信速度です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

ＲＳ－２３２Ｃの通信設定を参照します。

```
void func()
{
        BYTE    bPort, bByteSize, bFlowCtrl, bParity, bStopBits;
        DWORD   dwBaudRate;
        INT iComHandle;

        pPort = FIRST;
        if( CSCCommSetRs232cParam( pPort, L8, XCOFF, PNONE, S1, B9600 ) )
        {
                iComHandle = CSCCommInit();   /* ＲＳ－２３２Ｃによる初期化 */
                if( iComHandle != 0  )
                {
                        /* 通信処理 */

                        CSCCommGetRs232cParam( bPort, &bByteSize, &bFlowCtrl,
                                        &bParity, &bStopBits, &dwBaudRate );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
２２	NOT_SET_DEVICE	デバイスは設定されていません。

CSCCommSetCnetParam

機　能

Ｃ－ＮＥＴボードの設定

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommSetCnetParam( INT i0wnNode, UINT uiBufferBase,
                                                INT iIoSelect, INT iIrqSelect );
```

解　説

Ｃ－ＮＥＴの各パラメータを設定します。

<ＰＣ－９８０１用>

① ***wnNode*** は自局ノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***uiBufferBase*** はバッファベースアドレスです。

値	定　数	意　味
0xC000	BUF_MEM_C000	0xC0000～C3FFFを指定します。
0xC400	BUF_MEM_C400	0xC4000～C7FFFを指定します。
0xC800	BUF_MEM_C800	0xC8000～CBFFFを指定します。
0xCC00	BUF_MEM_CC00	0xCC000～CFFFFを指定します。
0xD000	BUF_MEM_D000	0xD0000～D3FFFを指定します。
0xD400	BUF_MEM_D400	0xD4000～D7FFFを指定します。
0xD800	BUF_MEM_D800	0xD8000～DBFFFを指定します。
0xDC00	BUF_MEM_DC00	0xDC000～DFFFFを指定します。

③ ***iIoSelec*** はＩ／Ｏアドレスです。

値	定　数	意　味
0	IO_0	0xC0D0,C0D2,C0D4,C0D6,C0D8,C0DA,C0DC,C0DEを指定します。
1	IO_1	0xC1D0,C1D2,C1D4,C1D6,C1D8,C1DA,C1DC,C1DEを指定します。
2	IO_2	0xC2D0,C2D2,C2D4,C2D6,C2D8,C2DA,C2DC,C2DEを指定します。
3	IO_3	0xC3D0,C3D2,C3D4,C3D6,C3D8,C3DA,C3DC,C3DEを指定します。

④ ***iIrqSelect*** は割り込みレベルです。

値	定　数	意　味
0	INT_0	IR3 (INT0)を指定します。
1	INT_1	IR5 (INT1)を指定します。
2	INT_2	IR6 (INT2)を指定します。
3	INT_3	IR9 (INT3)を指定します。
4	INT_4	IR10(INT4)を指定します。
5	INT_5	IR12(INT5)を指定します。
6	INT_6	IR13(INT6)を指定します。

<ＰＣ／ＡＴ用>

① ***wnNode*** は自局ノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

② ***BufferBase*** はバッファベースアドレスです。

値	定　数	意　味
0xC800	BUF_MEM_C800	0xC8000～CBFFFを指定します。
0xCC00	BUF_MEM_CC00	0xCC000～CFFFFを指定します。
0xD000	BUF_MEM_D000	0xD0000～D3FFFを指定します。
0xD400	BUF_MEM_D400	0xD4000～D7FFFを指定します。
0xD800	BUF_MEM_D800	0xD8000～DBFFFを指定します。
0xDC00	BUF_MEM_DC00	0xDC000～DFFFFを指定します。
0xE000	BUF_MEM_E000	0xE0000～E3FFFを指定します。
0xE400	BUF_MEM_E400	0xE4000～E7FFFを指定します。

③ ***iIoSelect*** はＩ／Ｏアドレスです。

値	定　数	意　味
0	IO_0	アドレス＝0x0330～0337を指定します。
1	IO_1	アドレス＝0x0338～033Fを指定します。
2	IO_2	アドレス＝0x0340～0347を指定します。
3	IO_3	アドレス＝0x0348～034Fを指定します。

④ ***iIrqSelect*** は割り込みレベルです。

値	定　数	意　味
0	INT_0	IRQ3 を指定します。
1	INT_1	IRQ4 を指定します。
2	INT_2	IRQ5 を指定します。
3	INT_3	IRQ7 を指定します。
4	INT_4	IRQ9 を指定します。
5	INT_5	IRQ10を指定します。
6	INT_6	IRQ11を指定します。
7	INT_7	IRQ12を指定します。
8	INT_8	IRQ14を指定します。
9	INT_9	IRQ15を指定します。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

Ｃ－ＮＥＴ（ＰＣ－９８０１の場合）の各パラメータを設定します。

```
void func()
{
        INT iComHandle;

        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＮＥＴによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

インターフェイス基板の設定と関数の引数は一致している必要があります。
CSCCommInit関数とペアで使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　数	意　味
－１	ARG_ERR	引数の値が異常です。
２０	CONNECT_ERROR	接続中（ｲﾝﾀｰﾌｪｲｽは初期化済み）です。

CSCCommGetCnetParam

機　能

Ｃ－ＮＥＴボードの参照

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommGetCnetParam( INT *iOwnNode, UINT *uiBufferBase,
                                                INT *iIoSelect, INT *iIrqSelect );
```

解　説

Ｃ－ＮＥＴの各パラメータを参照します。

① ***iOwnNode*** は自局ノード番号です。

② ***uiBufferBase*** はバッファベースアドレスです。

③ ***iIoSelect*** はＩ／Ｏアドレスです。

④ ***iIrqSelect*** は割り込みレベルです。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

Ｃ－ＮＥＴの通信設定を参照します。

```
void func()
{
        INT     iOwnNode, iIoSelect, iIrqSelect;
        UINT    uiBufferBase;
        INT    iComHandle;

        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
        {
                iComHandle = SCCommInit();     /* Ｃ－ＮＥＴによる初期化 */
                if( iComHandle != 0  )
                {
                        /* 通信処理 */

                        CSCCommGetCnetParam( &iOwnNode, &uiBufferBase,
                                                &iIoSelect, &iIrqSelect );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　数	意　味
２２	NOT_SET_DEVICE	デバイスは設定されていません。

CSCCommSetClinkParam

機　能

Ｃ－ＬＩＮＫポートの設定

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommSetClinkParam( BYTE bPort, BYTE bByteSize, BYTE bFlowCtrl,
                           BYTE bParity, BYTE bStopBits,
                           DWORD dwBaudRate, INT iOwnNode );
```

解　説

Ｃ－ＬＩＮＫの各パラメータを設定します。

① ***bPort*** はＲＳ－２３２Ｃのポートです。

値	定　数	意　味
1	FIRST	チャンネル０を使用します。
2	SECOND	チャンネル１を使用します。 （PC/ATのみ）

② ***bByteSize*** はデータ長です。

値	定　数	意　味
5	Ｌ５	５ｂｉｔｓです。
6	Ｌ６	６ｂｉｔｓです。
7	Ｌ７	７ｂｉｔｓです。
8	Ｌ８	８ｂｉｔｓです。

③ ***bFlowCtrl*** はＸフローコントロールです。

値	定　数	意　味
0	XCOFF	フロー制御を行いません。
4	XCON	フロー制御を行います。

④ ***bParity*** はパリティです。

値	定　数	意　味
0	PNONE	パリティなしです。
1	PODD	奇数パリティです。
2	PEVEN	偶数パリティです。

⑤ ***bStopBits*** はストップビットです。

値	定　数	意　味
0	Ｓ１	１ｂｉｔｓです。
1	Ｓ１５	１．５ｂｉｔｓです。
2	Ｓ２	２ｂｉｔｓです。

⑥ ***dwBaudRate*** は通信速度です。

値	定　数	意　味
300	B300	300bpsです。
600	B600	600bpsです。
1200	B1200	1200bpsです。
2400	B2400	2400bpsです。
4800	B4800	4800bpsです。
9600	B9600	9600bpsです。
19200	B19200	19200bpsです。
38400	B38400	38400bpsです。

⑦ ***iOwnNode*** は自局ノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

Ｃ－ＬＩＮＫによる通信を開始します。

```
void func()
{
        INT iComHandle;

        if( CSCCommSetClinkParam( FIRST, L8, XCOFF,
                                  PNONE, S1, B9600, NODE_0 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＬＩＮＫによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

WindowsNTでは、戻り値がTRUEであっても、意味をもちません。
CSCCommInit関数とペアで使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
２０	CONNECT_ERROR	接続中（ｲﾝﾀｰﾌｪｲｽは初期化済み）です。

CSCCommGetClinkParam

機　能

Ｃ－ＬＩＮＫポートの参照

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommGetClinkParam( BYTE bPort, BYTE *bByteSize, BYTE *bFlowCtrl,
                           BYTE *bParity, BYTE *bStopBits,
                           DWORD *dwBaudRate, INT *iOwnNode );
```

解　説

Ｃ－ＬＩＮＫの各パラメータを参照します。

① ***bPort*** はＲＳ－２３２Ｃのポートです。

② ****bByteSize*** はデータ長です。

③ ****bFlowCtrl*** はＸフローコントロールです。

④ ****bParity*** はパリティです。

⑤ ****bStopBits*** はストップビットです。

⑥ ****dwBaudRate*** は通信速度です。

⑦ ****iOwnNode*** は自局ノード番号です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

Ｃ－ＬＩＮＫの通信設定を参照します。

```
void func()
{
        BYTE    bPort, bByteSize, bFlowCtrl, bParity, bStopBits;
        DWORD   dwBaudRate;
        INT     iOwnNode;
        INT    iComHandle;

        bPort = FIRST;
        if( CSCCommSetClinkParam( bPort, L8, XCOFF,
                                          PNONE, S1, B9600, NODE_0 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＬＩＮＫによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommGetClinkParam( bPort, &bByteSize,
                                &bFlowCtrl,  &bParity, &bStopBits,
                                          &dwBaudRate, &iOwnNode );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

WindowsNTでは、戻り値がTRUEであっても、意味をもちません。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　数	意　味
－１	ARG_ERR	引数の値が異常です。
２２	NOT_SET_DEVICE	デバイスは設定されていません。

CSCCommSetParlinkParam

機　能

パラレル・リンクポートの設定

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommSetParlinkParam( SHORT sMachine, SHORT, sChannel );
```

解　説

パラレル・リンクの各パラメータを設定します。

① ***sMachine*** は使用するホストマシンです。

値	定　数	意　味
1	P_PCAT	IBM PCおよび互換機。
0	P_PC98	PC98シリーズおよび互換機。

② ***sChannel*** はパラレルポートのチャネル番号です。（IBM PCおよび互換機のみ）

値	定　数	意　味
0	P_CH_0	ポートアドレス378h～37Ahを指定します。
1	P_CH_1	ポートアドレス278h～27Ahを指定します。
2	P_CH_2	ポートアドレス3BCh～3BEhを指定します。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

パラレル・リンクによる通信を開始します。

```
void func()
{
        INT IComHandole;
        if ( CSCCommSetParlinkParam( P_PC98 , 0 ))   /*  0はダミー  /*
        {
                iComHandle = CSCCommInit(); /* パラレル・リンクによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

PC9800シリーズの場合、sChannelのデータは意味をもちません。
WindowsNTでは、戻り値がTRUEであっても、意味をもちません。
CSCCommInit関数とペアで使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
２０	CONNECT_ERROR	接続中（ｲﾝﾀｰﾌｪｲｽは初期化済み）です。

CSCCommGetParlinkParam

機　能

パラレル・リンクポートの参照

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommGetParlinkParam( SHORT *sMachine, SHORT, *sChannel );
```

解　説

パラレル・リンクの各パラメータを参照します。

① ***sMachine*** は使用するホストマシンです。

② ***sChannel*** はパラレルポートのチャネル番号です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

パラレル・リンクによる通信を参照します。

```
void func()
{
        INT IComHandole;
        SHORT sMachine, sChannel;

        if ( CSCCommSetParlinkParam( P_PCAT ,P_CH_0 ))
        {
                iComHandle = CSCCommInit(); /* パラレル・リンクによる初期化 */
                if( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommGetParlinkParam( &sMachine, &sChannel );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

PC9800シリーズの場合、*sChannelのデータは意味をもちません。
WindowsNTでは、戻り値がTRUEであっても、意味をもちません。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
２２	NOT_SET_DEVICE	デバイスは設定されていません。

CSCCommInit

機　能

インターフェイスの初期化（通信ハンドルの取得）

形　式

```
#include "csc_ctrl.h"
INT CSCCommInit( void );
```

解　説

インターフェイスの初期化するための制御関数です。
通信を開始する前に、必ずこの関数を呼び出す必要があります。

戻り値

値	定　数	意　味
0以外	なし	正常終了で、通信ハンドル
0	0	異常終了しました。

例

通信デバイスの設定関数を参照してください。同時に複数のインターフェイスを初期化（通信）する場合は、以下の例を参照してください。

```
void func()
{
        BYTE    bPort, bByteSize, bFlowCtrl, bParity, bStopBits;
        DWORD   dwBaudRate;

        INT     iOwnNode, iIoSelect, iIrqSelect;
        UINT    uiBufferBase;
        static INT    iComHandle1;
        static INT    iComHandle2;

        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE,S1, B9600 ) )
                    iComHandle1 = CSCCommInit();  /* RS-232Cによる初期化 */
        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400,IO_1, INT_5 ) )
                    iComHandle2 = CSCCommInit();  /* C-NETによる初期化   */

        if ( iComHandle1 != 0 && icomHandle2 != 0 )
        {
                /*
                 * RS-232Cポート０/C-NETによる通信処理
                 *
                 */


        }
        if ( iComHandle1 != 0 ) CSCCommFinish( iComHandle1 );
        if ( iComHandle2 != 0 ) CSCCommFinish( iComHandle2 );

}
```

留意事項

本関数をコールする前に、通信を行うデバイスの設定を行っておく必要があります。
各通信デバイスの設定を行う関数には以下があります。
また、設定に間違いがある場合、この関数から戻らない場合があります。
以下の設定関数とペアで使わなければなりません。

CSCCommSetRs232cParam
CSCCommSetCnetParam
CSCCommSetClinkParam(Windows95のみ)
CSCCommSetParlinkParam(Windows95のみ)

異常で戻り値が０の場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
－３	ALREADY_EXIST	既に当該ノード番号がＣ－ＮＥＴ上に存在しています。
６	LINE_BUSY	伝送路が２９０ミリ秒以上ビジィ状態であったため中断しました。
２０	CONNECT_ERROR	接続中（インターフェイスは初期化済み）です。
２３	INIT_ERROR	インターフェイスの初期化に失敗しました。
２４	DEVICE_ERROR	232C/CLINK同時通信できない。
４０	NO_DEVICE	サポートしていません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

ARG_ERRとは、デバイスの設定を行う関数で設定される引数が対象となります。

CSCCommFinish

機　能

インターフェイスの終了

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommFinish( INT iComHandle );
```

解　説

インターフェイスによる通信を終了するための制御関数です。
通信を終了する際には、必ずこの関数を呼び出す必要があります。
この時、初期化関数CSCCommInitの戻り値(通信ハンドル)が必要です。

① ***iComHandle*** は送受信処理ハンドルです。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

通信デバイスの設定関数を参照してください。

留意事項

ありません。

CSCCommGetNodeMap

機　能

Ｃ－ＮＥＴ上のノード・マップの取得

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommGetNodeMap( INT iDeviceID, BYTE *bMap );
```

解　説

Ｃ－ＮＥＴ上に存在するノードを調べて報告します。

① ***iDeviceID*** は通信ハンドルです。

② ****bMap*** はノード・マップ格納用配列のポインタです。
配列は、BYTE 型で 64 要素確保してください。
ノード番号をインデックスとして、配列内容が 0xFF の場合は該当ノードが存在し、0x00 の場合は存在しません。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

ノードマップを取得し、存在するノードを表示します。

```
void func( HWND hWnd   /* ウインドウのハンドル */ )
{
    INT         iComHandle;
    BYTE        bMap[64];
    INT         i, iCount;
    CHAR        cBuff[10];
    HDC         hDC;

    if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
    {
        iComHandle = CSCCommInit();   /* Ｃ－ＮＥＴによる初期化 */
        if( iComHandle != 0 )
        {
            if( CSCCommGetNodeMap( iComHandle, bMap ) )
            {
                hDC = GetDC( hWnd );
                for( i = 0, iCount = 0; i < 64; i++ )
                {
                    if( 0xff == bMap[i] )
                    {
                        wsprintf( cBuff, "NODE%d", i );
                        TextOut( hDC, 0, iCount*20,
                                              cBuff, strlen( cBuff ) );
                        iCount++;
                    }
                }
                ReleaseDC( hWnd, hDC );
            }
            CSCCommFinish( iComHandle );
        }
    }
}
```

留意事項

Ｃ－ＮＥＴのみ使用可能です。

戻り値が FALSE の場合、CSCCommGetErrorNo 関数でエラーの情報を取得することができます。

CSCCommGetErrorNo の戻り値は以下になります。

値	定　数	意　味
6	LINE_BUSY	伝送路がビジー状態です。
－2	NO_CONNECT	接続されていません。
８８８	HW_ERROR	インターフェースまたは関数に異常が発生しました。
９９９	UN_KNOWN	インターフェースまたは関数に未知の異常が発生しました。

CSCCommGets

機　能

データの受信

形　式

```
#include "csc_ctrl.h"
DWORD CSCCommGets( INT iComHandle, LPSTR lpszBlock,
                                    INT iMaxLength, INT iTrgNode, INT iRTime );
```

解　説

データを受信します。

① ***iComHandle*** は送受信処理ハンドルです。

② ***lpszBlock*** は受信データを格納するバッファのポインタです。

③ ***iMaxLength*** は受信要求データ長です。最大２０４８バイトです。

④ ***iTrgNode*** は送信元のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

受信したデータの長さを返します。

例

９０Ｘよりデータを受信し、表示します。

```
void func( HWND hWnd   /* ウインドウのハンドル */ )
{
        CHAR    cRecvBuff[2048];        /* 受信バッファ */
        DWORD   dwGetLen;
        INT     iComHandle;
        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
        {
                iComHandle = CSCCommInit();    /* Ｃ－ＮＥＴによる初期化 */
                if( iComHandle != 0 )
                {
                        dwGetLen= CSCCommGets( iComHandle,
                                                cRecvBuff, 2048, NODE_8, 30 );
                        cRecvBuff[dwPutLen] = '\0';
                        MessageBox( hWnd, cRecvBuff, "受信終了", MB_OK );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味をもちません。
ＲＳ－２３２Ｃ以外の場合、iMaxLengthのデータは意味をもちません。
戻り値が０の場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。
０	なし	受信できませんでした。

ＲＳ２３２Ｃの場合、受信処理後エラー情報は０（クリアー）されます。

CSCCommGets2

機　能

データの受信２

形　式

```
#include "csc_ctrl.h"
DWORD CSCCommGets2( INT iDeviceID, LPSTR lpszBlock,
                                    INT iMaxLength, INT *iTrgNode, INT iRTime );
```

解　説

データを受信します。

① ***iDeviceID*** は通信ハンドルです。

② ***lpszBlock*** は受信データを格納するバッファのポインタです。

③ ***iMaxLength*** は受信データ長です。最大２０４８バイトです。

④ ****iTrgNode*** は送信元のノード番号です。

⑤ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

受信したデータの長さを返します。

例

９０Ｘよりデータを受信し、表示します。

```
void func( HWND hWnd          /* ウインドウのハンドル */ )
{
        CHAR    cRecvBuff[2048];        /* 受信バッファ */
        DWORD   dwGetLen;
        INT     iComHandle;
        INT     iTrgNode;

        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＮＥＴによる初期化 */
                if( iComHandle != 0 )
                {
                        dwGetLen = CSCCommGets2( iComHandle, cRecvBuff,
                                                        2048, &iTrgNode, 30 );
                        cRecvBuff[dwGetLen] = '\0';
                        MessageBox( hWnd, cRecvBuff, "受信終了", MB_OK );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味を持ちません。
ＲＳ－２３２Ｃ以外の場合、iMaxLength のデータは意味を持ちません。
戻り値が FALSE の場合、CSCCommGetErrorNo 関数でエラーの情報を取得することができます。
CSCCommGetErrorNo の戻り値は以下になります。

値	定　数	意　味
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
８８８	HW_ERROR	インターフェースまたは制御関数に異常が発生しました。
９９９	UK_KNOWN	インターフェースまたは制御関数に未知の異常が発生しました。
０	なし	受信できませんでした。

CSCCommPuts

機　能

データの送信

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommPuts( INT iComHandle, LPSTR lpszBlock,
                                        LONG lLength, INT iTrgNode,INT iRTime );
```

解　説

データを送信します。

① ***iComHandle*** は送受信処理ハンドルです。

② ***lpszBlock*** は送信データが格納されているバッファのポインタです。

③ ***lLength*** は送信データ長です。最大２０４８バイトです。

④ ***iTrgNode*** は送信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

⑤ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例　　９０Ｘへデータを送信します。

```
void func( HWND hWnd   /* ウインドウのハンドル */ )
{
        CHAR    cSendBuff[2048];        /* 送信バッファ */
        LONG    lSendSize;
        BOOL    boPutRtn;
        INT     iComHandle:

        if( CSCCommSetClinkParam( FIRST, L8, XCOFF,
                                          PNONE, S1, B9600, NODE_0 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＬＩＮＫによる初期化 */
                if( iComHandle != 0 )
                {
                     strcpy( cSendBuff, "ABCDE" );
                     lSendSize = strlen( cSendBuff );
                     boPutRtn = CSCCommPuts(iComHandle,
                                          cSendBuff,lSendSize,NODE_8, 10);
                     if( boPutLtn )  MessageBox( hWnd,
                                          "正常終了", "送信結果",MB_OK );
                     else             MessageBox( hWnd,
                                         "異常終了", "送信結果", MB_OK );
                     CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項　ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味をもちません。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
１	NOT_READY	送信先のノードが受信準備されていませんでした。
２	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
６	LINE_BUSY	伝送路が２９０ミリ秒以上ビジィ状態であったため中断しました。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommOutCmd

機　能

システムコマンドの発行

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommOutCmd( INT iDeviceID, CHAR *s, INT iTrgNode, INT iRTime );
```

解　説

システムコマンドを発行します。

① ***iDeviceID*** は通信ハンドルです。

② ****s*** はコマンド列のポインタです。ラインターミネータは内部で自動的に付加します。

③ ***iTrgNode*** は送信先ノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

④ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例　　９０Ｘファイル操作モードに移行します。

```
void func( HWND hWnd   /* ウインドウのハンドル */ )
{
    CHAR       cCmdBuff[10];
    INT iComHandle;
    INT iTrgNode;       /* ＲＳ－２３２Ｃではダミー */

    if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
    {
        iComHandle = CSCCommInit();
                /* ＲＳ－２３２Ｃによる初期化 */
        if( iComHandle != 0 )
        {
            iTrgNode = 0;      /* ダミー */
            strcpy( cCmdBuff, "F" );
                /* システムコマンド：Ｆ発行 */
            if( CSCCommOutCmd( iComHandle, cCmdBuff, iTrgNode, 30 ) )
            {
                if( CMD_ACK == CSCCommCheckCmdAck
                                ( iComHandle, iTrgNode, 30 ) )
                {
                    if( CSCCommCheckSeq( iComHandle, PROMPT,
                                                30, iTrgNode ) )
                    {
                        MessageBox( hWnd, "ファイル操作モード移行成功",
                                                "処理結果", MB_OK );
                    }
                }
            }
            CSCCommFinish( iComHandle );
        }
    }
}
```

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味を持ちません。
戻り値が FALSE の場合、CSCCommGetErrorNo 関数でエラーの情報を取得することができます。
CSCCommGetErrorNo の戻り値は以下になります。

値	定　数	意　味
－１	ARG_ERR	引数の値が異常です。
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
１	NO_READY	送信先のノードが受信準備されていませんでした。
２	NO_ACK	送信先のノードがらアクノレッジが返ってこないので中止しました。
６	LINE_BUSY	伝走路が290ミリ以上ビジィ状態であったため中断しました。
１３	TIME_OUT	受信待機中、一定時間受信がありません。
１４	CMD_SIZE_ERROR	コマンドの長さが2048を越えています。
８８８	HW_ERROR	インターフェースまたは制御関数に異常が発生しました。
９９９	UK_KNOWN	インターフェースまたは制御関数に未知の異常が発生しました。

CSCCommCheckCmdAck

機　能

コマンド応答のチェック

形　式

```
#include "csc_ctrl.h"
CHAR CSCCommCheckCmdAck( INT iDeviceID, INT iTrgNode, INT iRTime );
```

解　説

システムコマンドの応答をチェックし、‘！’,‘％’,‘？’が返されるまで指定時間待ちます。

① ***iDeviceID*** は通信ハンドルです。

② ***iTrgNode*** は送信先ノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

③ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
‘！’	CMD_ACK	コマンド処理結果が正常終了しました。
‘％’	CMD_NAK	コマンド処理結果が異常終了しました。
‘？’	CMD_NON	未定義コマンドでした。
その他	なし	接続されていません。

例

CSCCommOutCmd を参照してください。

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味を持ちません。

CSCCommOutComand

機　能

システムコマンドの発行 ＆ コマンド応答のチェック

形　式

```
#include "csc_ctrl.h"
CHAR CSCCommOutComand( INT iComHandle, CHAR *s, INT iTrgNode, INT iRTime );
```

解　説

システムコマンドを発行して応答のチェックを行います。
この時、コマンドが返されるまで指定時間待ちます。

① ***iComHandle*** は送受信処理ハンドルです。

② ****s*** はコマンド列のポインタです。ラインターミネータは内部で自動的に付加します。

③ ***iTrgNode*** は送信先ノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

④ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
’！’	CMD_ACK	コマンド処理結果が正常終了しました。
’％’	CMD_NAK	コマンド処理結果が異常終了しました。
’？’	CMD_NON	未定義コマンドでした。
その他	なし	接続されていませんでした。

例

９０Ｘファイル操作モードに移行します。

```
void func( HWND hWnd   /* ウィンドウのハンドル */ )
{
        CHAR    cCmdBuff[10];
        CHAR    cCmdRtn;
        INT     iComHandle;

        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
        {
                iComHandle = CSCCommInit();   /* ＲＳ－２３２Ｃによる初期化 */
                if ( iComHandle != 0 )
                {
                        strcpy( cCmdBuff, "F" );/* システムコマンド：Ｆ発行 */
                        cCmdRtn = CSCCommOutComand( iComHandle,
                                                cCmdBuff, NODE_8, 30 );
                        if( ( CMD_ACK == cCmdRtn ) )
                        {
                                if( CSCCommCheckSeq( iComHandle,
                                                PROMPT, 30, NODE_8 ) )
                                        MessageBox( hWnd,
                         "ファイル操作モード移行成功", "処理結果", MB_OK );
                        }
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味をもちません。
９０Ｘのシステムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。

CSCCommCheckSeq

機　能

指定されたＡＣＫを指定時間待つ

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommCheckSeq( INT iComHandle, CHAR cc, INT iRTime, INT iTrgNode );
```

解　説

指定されたＡＣＫを指定時間待ちます。
９０１との通信を行う場合、一般的にシステムプロンプト('>')を指定します。
'>'はインクルードファイルで#define　PROMPT　'>'　として定義してあります。

① ***iComHandle*** は送受信処理ハンドルです。

② ***cc*** はコマンドＡＣＫを指定します。
システムプロンプトの場合、PROMPT('>')を指定します。
パラレル・リンクの場合、PROMPT('>')のみ有効です。

③ ***iRTime*** はリトライタイム（秒単位）です。

④ ***iTrgNode*** は送信元ノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

戻り値

値	定　数	意　味
１	TRUE	正常終了しました。
０	FALSE	異常終了しました。

例

CSCCommOutComandを参照してください。

留意事項

ＲＳ－２３２Ｃ、パラレル・リンクの場合、iTrgNode のデータは意味をもちません。
９０Ｘのシステムコマンドについては、９０Ｘ 操作説明書のシステムコマンドを参照してください。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　数	意　味
－２	NO_CONNECT	まだ接続されていません。
１３	TIME_OUT	一定時間受信がありませんでした。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommWatch

機　能

受信用イベントの検出

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommWatch( INT iComHandle, DWORD *pdwReceived );
```

解　説

受信の状態を調べて報告します。
受信が完了している場合にイベントを検出したことになります。

① ***iComHandle*** は送受信処理ハンドルです。

② ****pdwReceived*** は受信バイト数（必要ない場合、NULL代入します）
*pdwReceivedが0の場合、受信が有りません。
0以外の場合、受信が有ります。(RS-232Cの場合は受信バイト数で、C-NET/C-LINKの場合は１です)

戻り値

値	定　数	意　味
1	TRUE	イベントを検出しました。
0	FALSE	イベントが検出されませんでした。

例　　９０Ｘよりデータを受信します。

```
void func( INT iComHandle /*通信ハンドル*/,
                                HWND hWnd   /* ウィンドウのハンドル */ )
{
        CHAR            cRecvBuff[2048];
        CHAR            cBuff[100];
        DWORD           dwPutLen;
        DWORD           dwReceived;
        HDC             hDC;

        if( iComHandle != 0 )
        {
                hDC = GetDC( hWnd );

                strcpy( cBuff, "データの受信" );
                TextOut( hDC, 0, 0, cBuff, strlen( cBuff ) );

                while( !CSCCommWatch( iComHandle, &dwReceived ) )
                {
                        INT iErrorNumber;

                        iErrorNumber = CSCCommGetErrorNo();
                        if ( iErrorNumber != NO_EVENT )
                        {
                                strcpy( cBuff, "受信イベントが検出異常！" );
                                TextOut( hDC, 0, 20, cBuff, strlen( cBuff ) );
                                ReleaseDC( hWnd, hDC );
                                return; /* 異常終了 */
                        }
                        strcpy( cBuff, "イベントが検出されません" );
                        TextOut( hDC, 0, 20, cBuff, strlen( cBuff ) );
                }
                dwPutLen = CSCCommGets( iComHandle,
                                                cRecvBuff, 2048, NODE_8, 30 );
                if( dwPutLen )
                {
                        strcpy( cBuff, "データの受信終了" );
                        TextOut( hDC, 0, 20, cBuff, strlen( cBuff ) );
                }
                ReleaseDC( hWnd, hDC );
        }
}
```

留意事項

パラレル・リンクでは、サポートされていません。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－２	NO_CONNECT	まだ接続されていません。
３０	NO_EVENT	イベントが検出されませんでした。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommSendWatch

機　能

送信用イベントの検出

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommSendWatch( INT iComHandle, DWORD *pdwLeftBytes );
```

解　説

送信の状態を調べて報告します。
送信が完了している場合にイベントを検出したことになります。

① ***iComHandle*** は送受信処理ハンドルです。

② ****pdwLeftBytes*** は残る送信バイト数のポインタです（必要のない場合、NULLを代入します）。（C-NET/C-LINKの場合は送信完了すると０が代入されます）

戻り値

値	定　数	意　味
1	TRUE	イベントを検出しました。
0	FALSE	イベントが検出されませんでした。

例　　９０Ｘへデータを送信します。

```
void func( HWND hWnd  /* ウィンドウのハンドル */ )
{
        CHAR    cSendBuff[2048];
        CHAR    cBuff[100];
        LONG    lSendSize;
        BOOL    boPutsRtn;
        HDC     hDC;
        INT     iComHandle;
        DWORD   dwLeftBytes;

        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
        {
                iComHandle = CSCCommInit();   /* ＲＳ－２３２Ｃによる初期化 */
                if( iComHandle != 0 )
                {
                        hDC = GetDC( hWnd );
                        strcpy( cBuff, "データの送信" );
                        TextOut( hDC, 0, 0, cBuff, strlen( cBuff ) );

                        strcpy( cSendBuff, "ABCDE" );
                        lSendSize = strlen( cSendBuff );
                        boPutsRtn = CSCCommPuts( iComHandle, cSendBuff,
                                                lSendSize,NODE_8, 30 );
                        if( boPutsRtn )
                        {
                              while( !CSCCommSendWatch( iComHandle,
                                                &dwLeftBytes ) )
                              {
                                      INT iErrorNumber;

                                      iErrorNumber = CSCCommGetErrorNo();
                                      if ( iErrorNumber != NO_EVENT)
                                                break;/* 異常終了 */
                                      strcpy( cBuff, "データの送信中" );
                                      TextOut( hDC, 0, 20, cBuff,
                                                strlen( cBuff ) );
                                 }
                        }
                        strcpy( cBuff, "データの送信終了" );
                        TextOut( hDC, 0, 20, cBuff, strlen( cBuff ) );

                        ReleaseDC( hWnd, hDC );
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

パラレル・リンクでは、サポートされていません。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－２	NO_CONNECT	まだ接続されていません。
１	NOT_READY	送信先のノードが受信準備されていませんでした。
２	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
６	LIVE_BUSY	伝送路が２９０ミリ秒以上ビジィ状態であったため中断しました。
３０	NO_EVENT	イベントが検出されませんでした。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommLoad

機　能

ファイルの送信

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommLoad( INT iComHandle, INT iMode, INT iTrgNode,
                                HANDLE hFile, void (*func)(DWORD totalsize,
                                                    DWORD size),  INT iRTime );
```

解　説

９０１用のファイルを９０１に送信します。

① ***iComHandle*** は送受信処理ハンドルです。

② ***iMode*** はファイルモードです。

値	定　数	意　味
0	SYSTEMFILE	システムファイルの送信を指定します。
1	USERFILE	ユーザファイルの送信を指定します。

③ ***iTrgNode*** は送信先のノード番号です。

値	定　数	意　味
0	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

④ ***hFile*** はオープンされたファイルのハンドルです。

⑤ ****func*** はユーザ定義の関数のポインタです。
この関数の引数として

totalsize
　送信データの総バイト数（最初の呼出時のみ有効、以降は０）
size
　今回送信したバイト数

を渡します。

⑥ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例　USER.BIN を９０Ｘへ送信します。

```
void func( HWND hWnd   /* ウィンドウのハンドル */ )
{
        HANDLE          hFile;
        BOOL            boRtn
        INT             iComHandle;
        static SECURITY_ATTRIBUTES
                        sa = {  /* Win32 SECURITY_ATTRIBUTES構造体       */
                sizeof(SECURITY_ATTRIBUTES),
                                /* Win95ならsa.nLength =sizeof()だけで可 */
                NULL,
                TRUE
        };
        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
        {
                iComHandle = CSCCommInit();   /* ＲＳ－２３２Ｃによる初期化 */
                if( iComHandle != 0 )
                {
                        hFile = CreateFile(    "a:\\csc901\\USER.BIN",
                                               GENERIC_READ,
                                               0,
                                               &sa,
                                               OPEN_EXISTING,
                                               FILE_ATTRIBUTE_NORMAL,
                                               0 );
                        if( hFile != (HANDLE)NULL )
                        {
                                boRtn = CSCCommLoad( iComHandl, USERFILE,
                                                     NODE_8, hFile, NULL, 30 );
                                MessageBox(    hWnd,
                                               boRtn==TRUE ? "成功":"失敗",
                                               "ファイル送信", MB_OK );
                                CloseHandle( hFile );
                        }
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

WindowsNTでのファイル転送はセキュリテイをチェックされます。
９０X側のユーザプログラムでファイルを受信するには、９０X 基本ライブラリのLib_fcopyを使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－２	NO_CONNECT	まだ接続されていません。
－４	NOT_COMPLETE	まだ送信が完了していません。
－５	SEND_TO_ME	送信先が自局になっています。
１	NOT_READY	送信先のノードが受信準備されていませんでした。
２	NO_ACK	送信先のノードからアクノレッジが返ってこないので中止しました。
６	LINE_BUSY	伝送路が２９０ミリ秒以上ビジィ状態であったため中断しました。
１０	FILE_SIZE_ERROR	ファイルのサイズが短いです。
１３	TIME_OUT	送信タイムアウトエラー。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommUpLoad

機　能

ファイルの受信

形　式

```
#include "csc_ctrl.h"
BOOL CSCCommUpLoad( INT iComHandle, INT iTrgNode, HANDLE hFile,
                    void (*func)(DWORD totalsize, DWORD size), INT iRTime );
```

解　説

９０Ｘ用のファイルを９０Ｘから受信します。

① ***iComHandle*** は送受信処理ハンドルです。

② ***iTrgNode*** は送信先のノード番号です。

値	定　数	意　味
０	NODE_0	ノード番号０を指定します。
・	・	・
６３	NODE_63	ノード番号６３を指定します。

③ ***hFile*** はオープンされたファイルのハンドルです。

④ ****func*** はユーザ定義の関数のポインタです。
この関数の引数として

totalsize
　送信データの総バイト数（最初の呼出時のみ有効、以降は０）
size
　今回送信したバイト数

を渡します。

⑤ ***iRTime*** はリトライタイム（秒単位）です。

戻り値

値	定　数	意　味
1	TRUE	正常終了しました。
0	FALSE	異常終了しました。

例

TEST.BIN を９０Ｘから受信します。

```
void func()
{
        HANDLE          hFile;
        BOOL            boRtn
        INT             iComHandle;
        static SECURITY_ATTRIBUTES
                        sa = {  /* Win32 SECURITY_ATTRIBUTES構造体       */
                sizeof(SECURITY_ATTRIBUTES),
                                /* Win95ならsa.nLength = sizeof()だけで可*/
                NULL,
                TRUE
        };

        if( CSCCommSetCnetParam( NODE_0, BUF_MEM_C400, IO_1, INT_5 ) )
        {
                iComHandle = CSCCommInit();   /* Ｃ－ＮＥＴによる初期化 */
                if ( iComHandle != 0 )
                 {
                        hFile = CreateFile(    "c:\\csc901\\TEST.BIN",
                                               GENERIC_WRITE,
                                               0,
                                               &sa,
                                               CREATE_ALWAYS,
                                               FILE_ATTRIBUTE_NORMAL,
                                               0 );
                        if( hFile != (HANDLE)NULL )
                        {
                                boRtn = CSCCommUpLoad( iComHandle,
                                                NODE_8, hFile, NULL, 30 );
                                if( boRtn )
                                        MessageBox( hWnd,
                                         "成功", "ファイル送信", MB_OK );
                                else
                                        MessageBox( hWnd,
                                         "失敗", "ファイル送信", MB_OK );
                                CloseHandle( hFile );
                        }
                        CSCCommFinish( iComHandle );
                }
        }
}
```

留意事項

WindowsNTでのファイル転送はセキュリテイをチェックされます。
９０Ｘ側のユーザプログラムでファイルを送信するには９０Ｘ 基本ライブラリLib_fcopyを使用します。
戻り値がFALSEの場合、CSCCommGetErrorNo関数でエラーの情報を取得することができます。
CSCCommGetErrorNoの戻り値は以下になります。

値	定　　数	意　　味
－２	NO_CONNECT	まだ接続されていません。
１２	FILE_WRITE_ERROR	ファイル書き込みエラーが発生しました。
１３	TIME_OUT	受信待機中、一定時間受信がありませんでした。
８８８	HW_ERROR	インターフェイスまたは制御関数に異常が発生しました。
９９９	UN_KNOWN	インターフェイスまたは制御関数に未知の異常が発生しました。

CSCCommGetErrorNo

機　能

最新のエラー情報の取得

形　式

```
#include "csc_ctrl.h"
INT CSCCommGetErrorNo( void );
```

解　説

エラーが発生した場合、直後の呼び出しでエラーの内容を知ることができます。
次のエラー発生で前回のエラー情報は失われます。

戻り値

エラー内容によります。

例

エラーが発生した場合、エラー内容を取得します。

```
void func ( HWND hWnd /*ウィンドウのハンドル*/   )
{
        INT     iComHandle;
        INT     iErrorInfo;
        HDC     hDC;

        Hdc = GetDCChWnd( );
        if( CSCCommSetRs232cParam( FIRST, L8, XCOFF, PNONE, S1, B9600 ) )
        {
/               iComHandle = CSCCommInit();
                if ( iComHandle != 0 )
                {
                        /* 通信処理 */

                        CSCCommFinish( iComhandle );
                }
                else
                {
                        iErrorInfo = CSCCommGetErrorNo( )
                        /* エラー内容の表示等の処理*/
                }
                ReleaseDC( hWnd,hDC );
        }
}
```

留意事項

ライブラリによりエラー情報を取得できないものもあります。

３．サンプルプログラム

本ライブラリを使用したプログラム例として、サンプルが

＜ＰＣ－９８用＞

\cscsmp1\cscsmp9 .c ・・・・・・・・・・・・・・・・ ソースファイル
\cscsmp1\cscsmp9 .rc ・・・・・・・・・・・・・・・・ リソースファイル
\cscsmp1\resource.h ・・・・・・・・・・・・・・・・ ヘッダーファイル
\cscsmp1\cscsmp9 .mak ・・・・・・・・・・・・・・・ メイクファイル
\cscsmp1\cscsmp9 .exe ・・・・・・・・・・・・・・・ 実行ファイル
(\cscsmp1\CSC_C98.LIB ・・・・・・・・・・・・・・・ 通信用ＬＩＢファイル)
(\cscsmp1\CSC_CTRL.H ・・・・・・・・・・・・・・・・ ＬＩＢ用インクルードファイル)
(\cscsmp1\CSC_C98.DLL ・・・・・・・・・・・・・・・ 通信用ＤＬＬファイル)

＜ＰＣ／ＡＴ用＞

\cscsmp9\cscsmpl .c ・・・・・・・・・・・・・・・・ ソースファイル
\cscsmp9\cscsmpl .rc ・・・・・・・・・・・・・・・・ リソースファイル
\cscsmp9\resource.h ・・・・・・・・・・・・・・・・ ヘッダーファイル
\cscsmp9\cscsmpl .mak ・・・・・・・・・・・・・・・ メイクファイル
\cscsmp9\cscsmpl .exe ・・・・・・・・・・・・・・・ 実行ファイル
(\cscsmp9\CSC_COMM. LIB ・・・・・・・・・・・・・・ 通信用ＬＩＢファイル)
(\cscsmp9\CSC_CTRL. H ・・・・・・・・・・・・・・・ ＬＩＢ用インクルードファイル)
(\cscsmp9\CSC_COMM. DLL ・・・・・・・・・・・・・・ 通信用ＤＬＬファイル)

としてあります。
LIB,Hファイルは、サンプルプログラムの作成時に統合環境に組み込むため、DLLファイルは、実行ファイルを動作する上で必要なため、同一フォルダに存在します。
このプログラムは、パソコンから９０ＸのＦＳ０にファイルをコピーするものです。

起動

ファイルマネージャ（エクスプローラなど）から実行ファイルを開きます。

使い方

このプログラムには、以下のメニューがあります。

[File]-[Download...]

このメニューを選択すると、ファイル選択用ダイアログボックスが表示されます。
ファイル名を選択して、[開く]ボタンをクリックするとダウンロード（パソコンから９０Ｘへコピー）を開始します。
ダウンロード終了後、メッセージが表示されます。

[File]-[Exit]

このメニューを選択すると、本プログラムを終了します。

[Environment]-[Device...]

このメニューを選択すると、通信デバイス選択用ダイアログボックスが表示されます。
デバイスを選択後、右下のボタンをクリックするとパラメータを選択するダイアログボックスが表示されます。
全て[ＯＫ]ボタンをクリックすることにより、設定が有効になります。

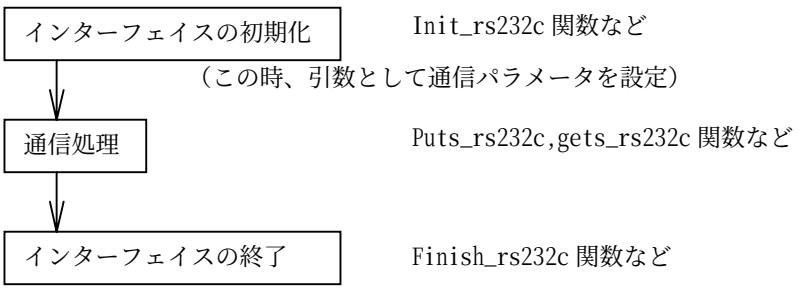
通信設定完了後、ダウンロードを開始してください。

４．補足説明

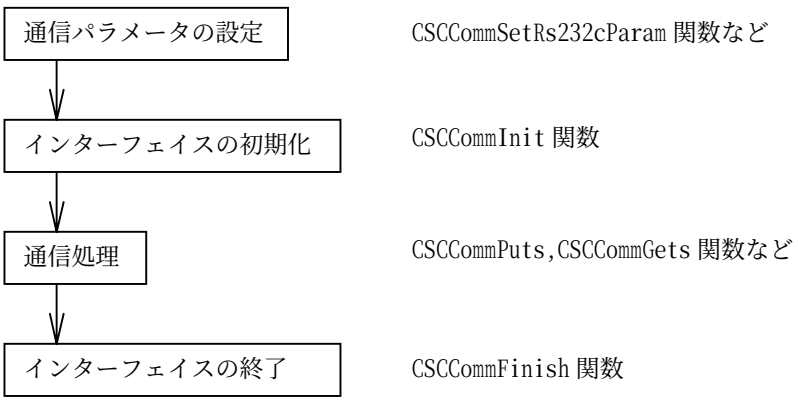
Ｗｉｎｄｏｗｓ９５／ＮＴ対応ホスト通信ライブラリ（以後Ｗｉｎｄｏｗｓ）とホストＣ－ＮＥＴライブラリ、ホストＣ－ＬＩＮＫライブラリ、ホストパラレル・リンクライブラリ（以後ＤＯＳ）との相違点について述べたものです。

4.1　通信処理（コーディング）のながれ

[ＤＯＳ版]



[Ｗｉｎｄｏｗｓ版]



4.2　Windows 版ライブラリの紹介と DOS 版との比較

＜対応するライブラリ＞

○インターフェイスの初期化

[ＤＯＳ] RS-232C　: init_rs232c
C-NET　: C_NET_Init
C-LINK　: init_clink
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : init_par_link
引数に通信パラメータ（ボーレートなど）を与えます。

[Ｗｉｎｄｏｗｓ] CSCCommInit
事前に通信パラメータを与えておく必要があります。
通信パラメータを与える関数として以下があります。
RS-232C　: CSCCommSetRs232cParam
C-NET　: CSCCommSetCnetParam
C-LINK　: CSCCommSetClinkParam（Windows95 のみ）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : CSCCommSetParlinkParam（Windows95 のみ）

○インターフェイスの終了

[ＤＯＳ] RS-232C　: finish_rs232c
C-NET　: C_NET_Finish
C-LINK　: finish_clink
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : finish_par_clink

[Ｗｉｎｄｏｗｓ] CSCCommFinish

○データの送信

[ＤＯＳ] RS-232C　: puts_rs232c
C-NET　: puts_cnet
C-LINK　: puts_clink（WindowsNT では相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : puts_par_clink（WindowsNT では相当関数は、ありません）
戻り値として、処理結果を返します。

[Ｗｉｎｄｏｗｓ] CSCCommPuts
ＤＯＳ版と引数、戻り値に変更あります。
戻り値として、TRUE か FALSE を返します。

○データの受信

[ＤＯＳ] RS-232C　: gets_rs232c
C-NET　: gets_cnet
C-LINK　: gets_clink（WindowsNT では相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : gets_par_link（WindowsNT では相当関数は、ありません）
戻り値として、処理結果を返します。

[Ｗｉｎｄｏｗｓ] CSCCommGets
ＤＯＳ版と引数、戻り値に変更あります。
戻り値として、受信した長さを返します。

○ファイルの送信

[ＤＯＳ] RS-232C　: load_rs232c
C-NET　: cload
C-LINK　: load_clink（WindowsNTでは相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : load_par_link（WindowsNTでは相当関数は、ありません）

[Ｗｉｎｄｏｗｓ] CSCCommLoad
ＤＯＳ版と引数に変更あります。

○ファイルの受信

[ＤＯＳ] RS-232C　: upload_rs232c
C-NET　: cupload
C-LINK　: upload_clink（WindowsNTでは相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : upload_par_link（WindowsNTでは相当関数は、ありません）

[Ｗｉｎｄｏｗｓ] CSCCommUpLoad
ＤＯＳ版と引数に変更あります。

○システムコマンドの発行＆コマンド応答チェック

[ＤＯＳ] RS-232C　: outcmd_rs232c + checkcmdack_rs232c
C-NET　: outccmd + checkccmdack
C-LINK　: outcmd_clink + checkcmdack_clink
（WindowsNTでは相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : upload_par_link + checkcmdack_par_link
（WindowsNTでは相当関数は、ありません）

[Ｗｉｎｄｏｗｓ] CSCCommOutComand
ＤＯＳ版の２つの関数をまとめて１つにした関数です。
よって、引数、戻り値に変更あります。

○指定されたＡＣＫを指定時間待つ（プロンプトチェック）

[ＤＯＳ] RS-232C　: checkprmpt_rs232c
C-NET　: checkcprmpt
C-LINK　: checkprmpt_clink（WindowsNTでは相当関数は、ありません）
ﾊﾟﾗﾚﾙ･ﾘﾝｸ : checkprmpt_par_link（WindowsNTでは相当関数は、ありません）

[Ｗｉｎｄｏｗｓ] CSCCommCheckSeq
ＤＯＳ版と引数に変更あります。（パラレル・リンクでは使い方に制限があります。）

○受信用イベントの検出(受信ステータスの取得)

[ＤＯＳ] C-NET　: C_NET_Receive_Status

[Ｗｉｎｄｏｗｓ] CSCCommWatch
ＤＯＳ版と引数,戻り値に変更あります。
全てのデバイスに有効です。（ただし、使い方に制限があります）
パラレル・リンクでは対応していません。

○送信用イベントの検出(受信ステータスの取得)

[ＤＯＳ] C-NET　: C_NET_Send_Status

[Ｗｉｎｄｏｗｓ] CSCCommSendWatch
ＤＯＳ版と引数,戻り値に変更あります。
全てのデバイスに有効です。（ただし、使い方に制限があります）
パラレル・リンクでは対応していません。

Windows95/NT版　ホスト通信
＜新規ライブラリ＞

○ＲＳ－２３２Ｃポートの設定
CSCCommSetRs232cParam

○ＲＳ－２３２Ｃポートの参照
CSCCommGetRs232cParam

○Ｃ－ＮＥＴボードの設定
CSCCommSetCnetParam

○Ｃ－ＮＥＴボードの参照
CSCCommGetCnetParam

○Ｃ－ＬＩＮＫポートの設定
CSCCommSetClinkParam

○Ｃ－ＬＩＮＫポートの参照
CSCCommGetClinkParam

○パラレル・リンクポートの設定
CSCCommSetParlinkParam

○パラレル・リンクポートの参照
CSCCommGetParlinkParam

○通信異常状態を取得
CSCCommGetErrorNo

＜その他＞

Ｃ－ＮＥＴ基本ライブラリ、割り込み関連ライブラリなど、サポートしていないものもあります。

通信速度

通信速度はＲＳ－２３２Ｃを１とした場合Ｃ－ＮＥＴが６０～８０倍(4Mbps)，Ｃ－ＬＩＮＫは、パソコンの性能にも依存しますが５～８倍(50Kbps～100Kbps)、パラレル・リンクもパソコンの性能にも依存しますが２５～４０倍になります。

各通信機器の通信速度例として、ホストのハードディスク上のファイルをＣＳＣ９０ＸのＦＳ０にファイルコピーした場合は以下のようになります。

①

ホスト　　ＰＣ９８２１Ａｐ２（４８６ＤＸ２，６６ＭＨｚ）
ＩＤＥハードディスク，ディスクキャッシュなし
ＯＳ：ＭＳ－ＤＯＳ　Ｖｅｒ．６．２

９０１ｃ　　ＦＳ０は、フラッシュ・ファイル・カード

ファイル　　１００Ｋｂｙｔｅ

通信機器	時　間
ＲＳ－２３２Ｃ(9600bps)	約１２９秒
Ｃ－ＬＩＮＫ	約　１５秒
Ｃ－ＮＥＴ	約　　３秒

・時間は、ホストのディスクアクセスを含みます。

②

ホスト　　DELL　Optiplex GXL 5133（Pentium™プロセッサ，133MHz）
エンハンスドＩＤＥハードディスク、セカンドキャッシュ　256Kbyte
ＯＳ：Ｗｉｎｄｏｗｓ９５

９０１ｃ　　ＦＳ０は、フラッシュ・ファイル・カード

ファイル　　１００Ｋｂｙｔｅ

通信機器	時　間
ＲＳ－２３２Ｃ(9600bps)	約１１８秒
Ｃ－ＬＩＮＫ	約　１５秒
Ｃ－ＮＥＴ	約　　３秒

・時間は、ホストのディスクアクセスを含みます。

③

ホスト　　ＰＣ９８２１Ｘａ７（Pentium™プロセッサ，75MHz）
エンハンスドＩＤＥハードディスク
ＯＳ：ＭＳ－ＤＯＳ　Ｖｅｒ．６．２

９０２ａ　　ＦＳ０は、フラッシュ・ファイル・カード

ファイル　　１００Ｋｂｙｔｅ

通信機器	時　間
ＲＳ－２３２Ｃ(9600bps)	約１３０秒
Ｃ－ＬＩＮＫ	約　１２秒
Ｃ－ＮＥＴ	約　　２秒
パラレル・リンク	約　　３秒

・時間は、ホストのディスクアクセスを含みます。

④

ホスト　　DELL　Optiplex GXL 5133（Pentium™プロセッサ，133MHz）
エンハンスドＩＤＥハードディスク、セカンドキャッシュ　256Kbyte
ＯＳ：Ｗｉｎｄｏｗｓ９５

９０２ａ　　ＦＳ０は、フラッシュ・ファイル・カード

ファイル　　１００Ｋｂｙｔｅ

通信機器	時　間
ＲＳ－２３２Ｃ(9600bps)	約１１７秒
Ｃ－ＬＩＮＫ	約　１２秒
Ｃ－ＮＥＴ	約　　２秒
パラレル・リンク	約　　３秒

・時間は、ホストのディスクアクセスを含みます。

⑤

ホスト　　ＰＣ／ＡＴ互換機（AMD K6プロセッサ，233MHz）

９０２ＳＴ　ＦＳ０は、フラッシュ・ファイル・カード

ファイル　　１００Ｋｂｙｔｅ

通信機器	時　間
ＲＳ－２３２Ｃ (9600bps)	約１１７秒
ＲＳ－２３２Ｃ(19200bps)	約　５９秒
ＲＳ－２３２Ｃ(38400bps)	約　３０秒
Ｃ－ＬＩＮＫ	約　１２秒
Ｃ－ＮＥＴ	約　　１秒
パラレル・リンク	約　　２秒

・時間は、ホストのディスクアクセスを含みます。

９０Ｘシリーズ

９０Ｘ　ホスト通信ライブラリ説明書

２０００年７月第３版第２刷発行

発行所　　株式会社ファースト

本　　社　〒242-0001　神奈川県大和市下鶴間２７９１－５

ユーザ・サポート　　**FAX　046-272-8692**　　TEL　046-272-8691

E-mail : support@fast-corp.co.jp

B-000743